Register File Design with 32x8-bit Word SRAM Memory Arrayusing 5-bit Address on Cadence Virtuoso 180nm Technology

Rishab Vyas Mahendra
02008386

University of Massachusetts Lowell
ECE Department

Table of Contents

1. Introduction

In the world of computer design, the Register File is a crucial part that acts as a fast storage space for holding and retrieving temporary data. Among the different technologies used for creating register files, Static Random Access Memory (SRAM) plays a significant role, offering benefits in terms of speed, simplicity, and efficiency.

SRAM is known for its ability to keep data without needing constant refreshing, forming the base of the Register File and allowing rapid access to stored information for the processor. This dynamic interaction between SRAM cells and the Register File is essential for making data manipulation within a processor quick and efficient.

2. Problem Formulation

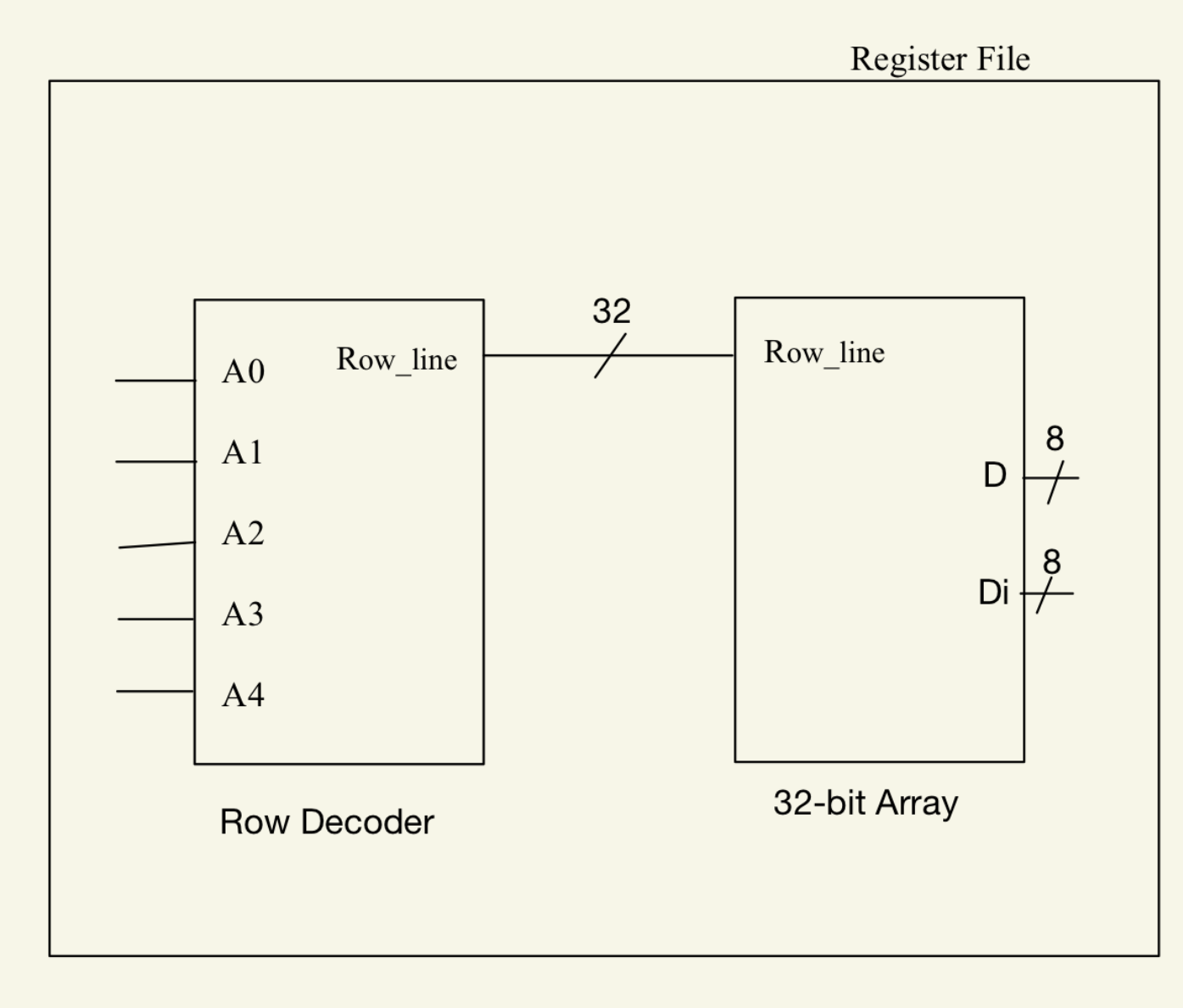
The primary objective of this project was to engineer a 32x8 SRAM Memory Array Register file, drawing inspiration from an IEEE paper that focused on an 8x8 SRAM Memory Array. In our design, the Register File comprises a 32x8 SRAM Memory Array, consisting of 32 words, with each word composed of 8 bits. Each individual bit is constructed using a single-cell 6T SRAM configuration. The critical functionality of reading and writing to the Memory Array is facilitated through the integration of a Row decoder.

The 5:32 Row decoder is used in determining the specific word to which data is written or from which data is read. This decoding process is driven by a 5-bit address input. The Row decoder itself is constructed using a 5-input AND gate, acting as the decision-making component in the memory access operations.

The Main components of Project are

- Inverter (12u/6u and 1.5u /1.5 u)
- 6T SRAM Cell
- 8-bit SRAM
- 32-bit Array
- 5 input AND gate
- Row Decoder
- Register File

These components collectively form a robust and efficient Register File, incorporating advancements inspired by the referenced IEEE paper. The focus on scalability from an 8x8 to a 32x8 SRAM Memory Array underscores the project's commitment to enhancing data storage and retrieval capabilities within the specified architectural framework.

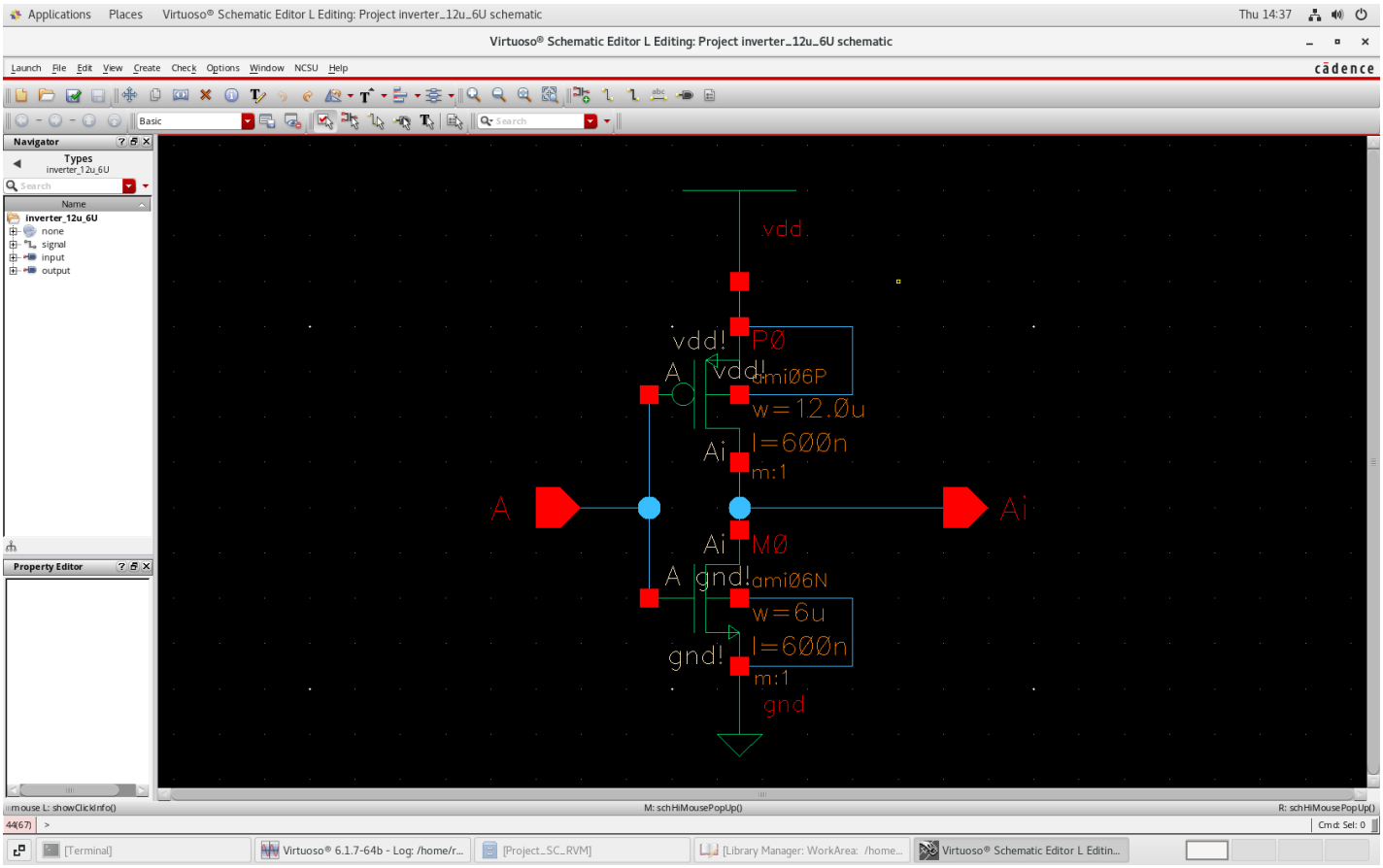


Project High Level Schematic

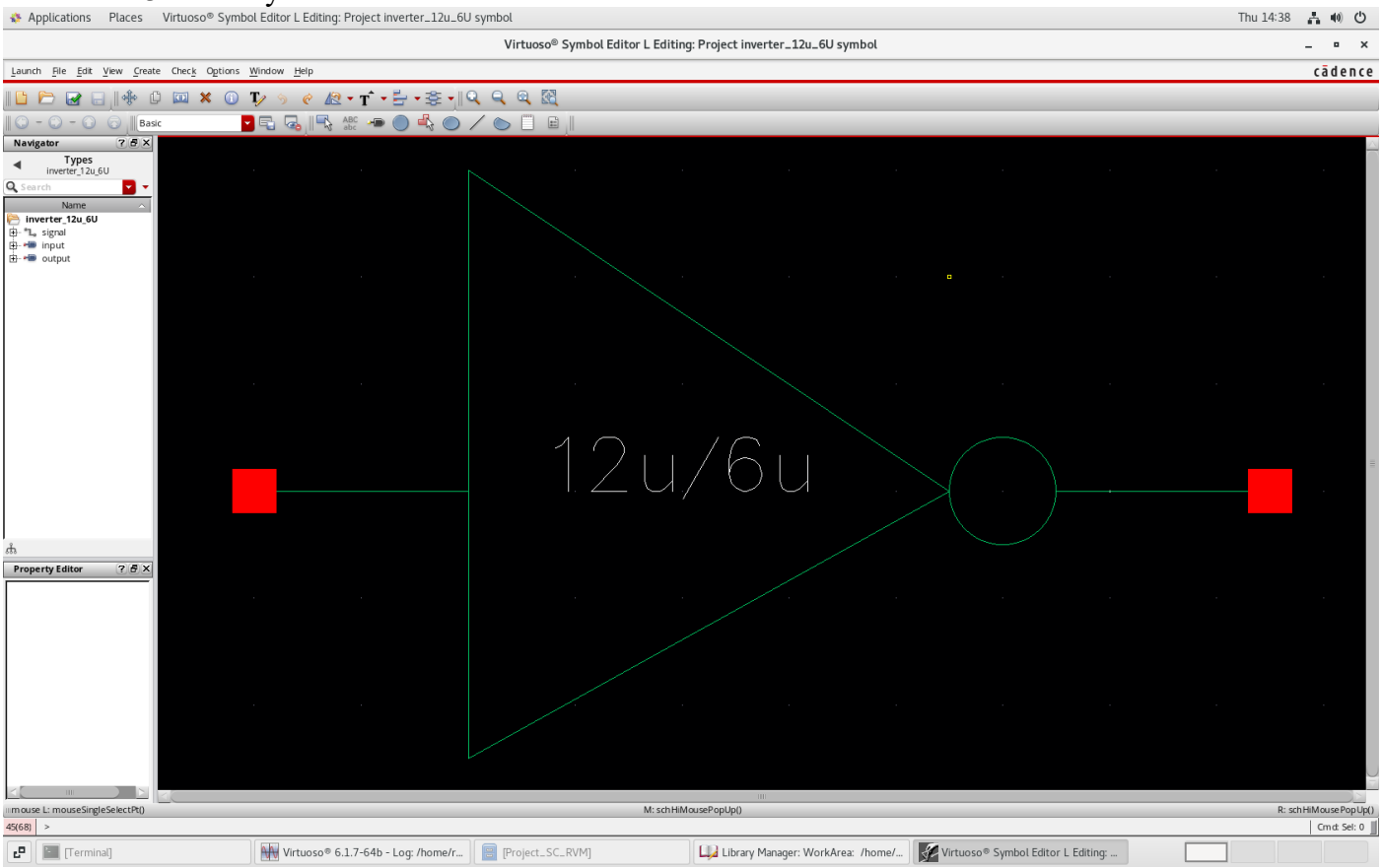
3. CMOS Design

3.1 Inverter (12u/6 u)

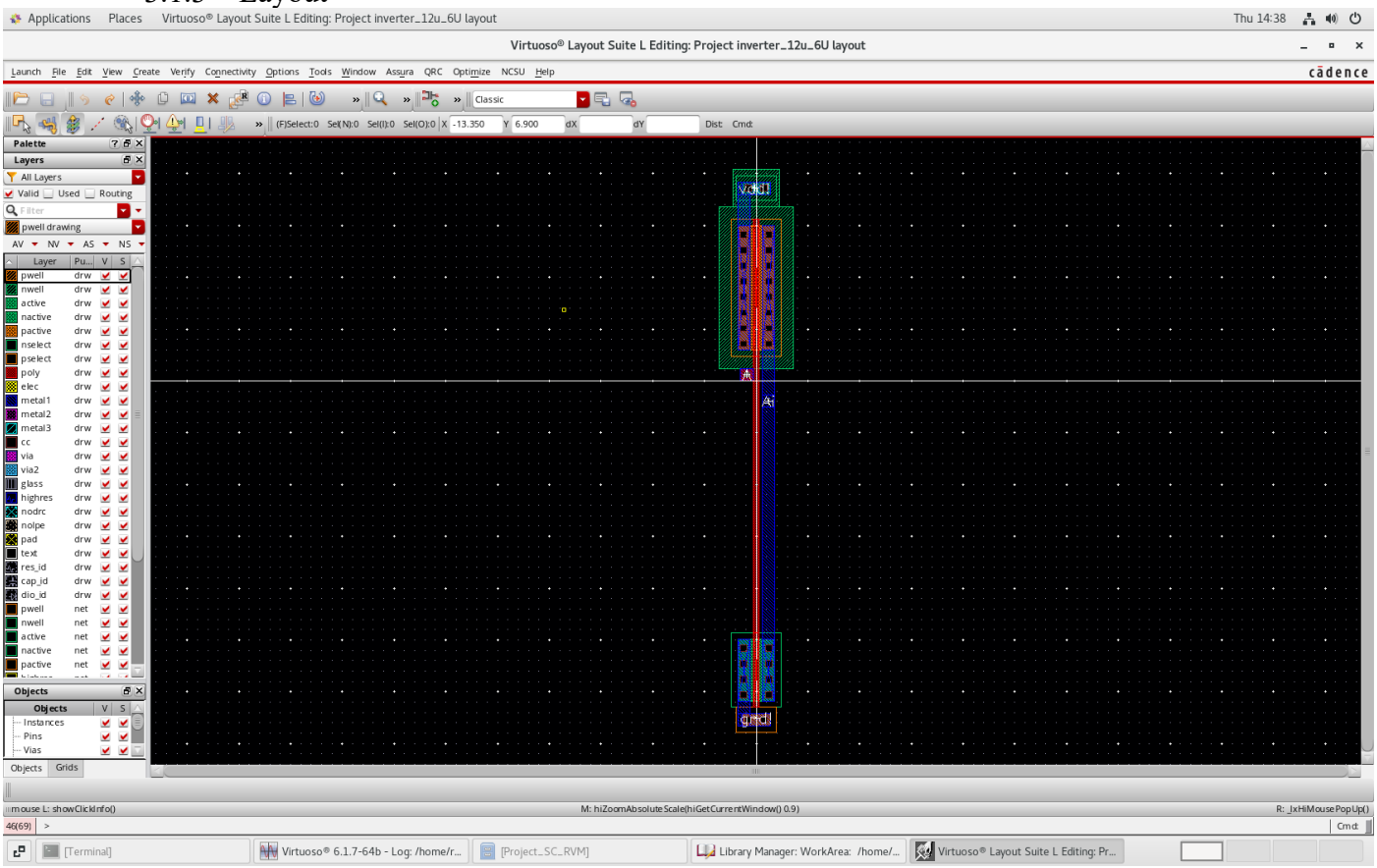
3.1.1 Schematic



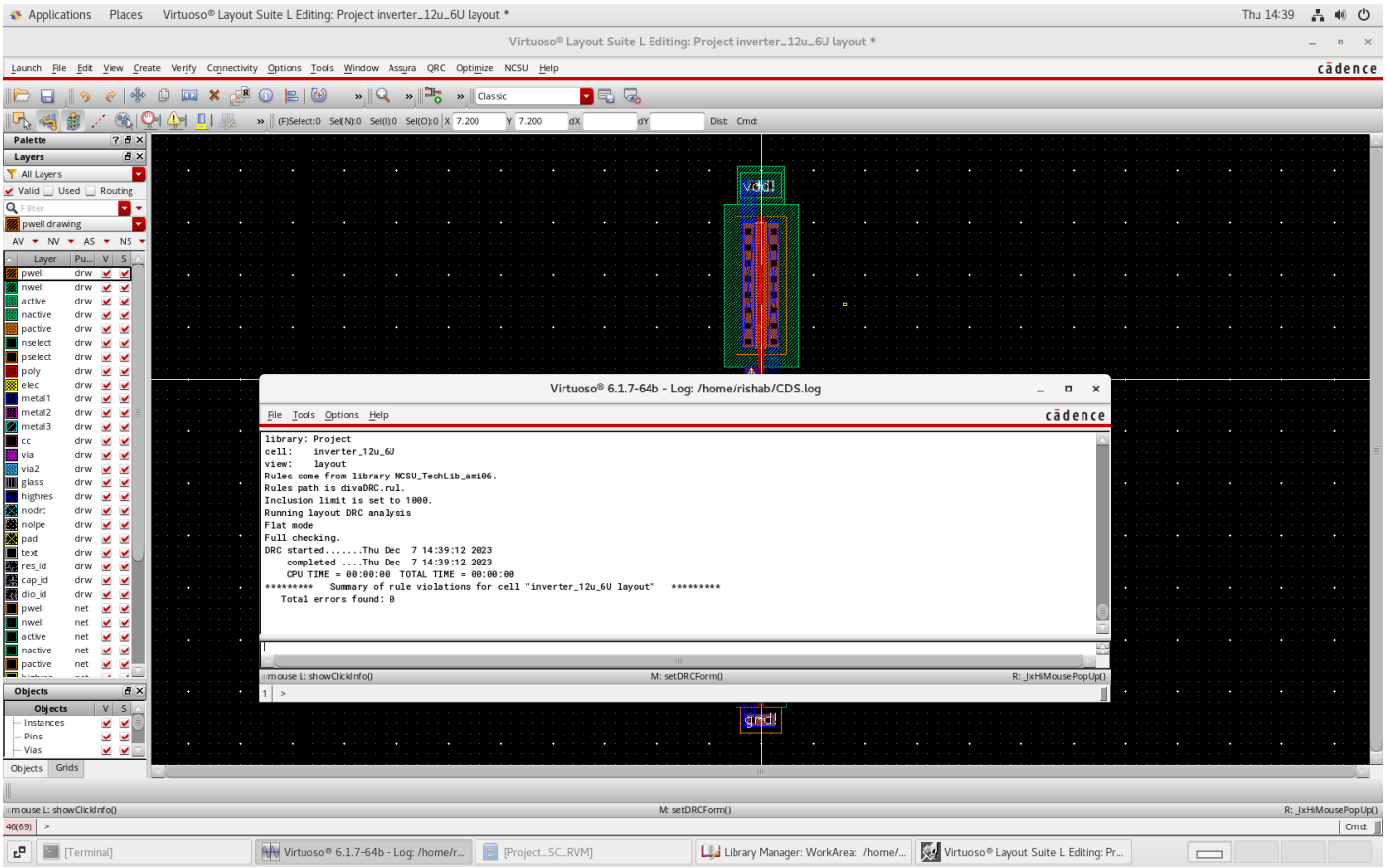
3.1.2 Symbol



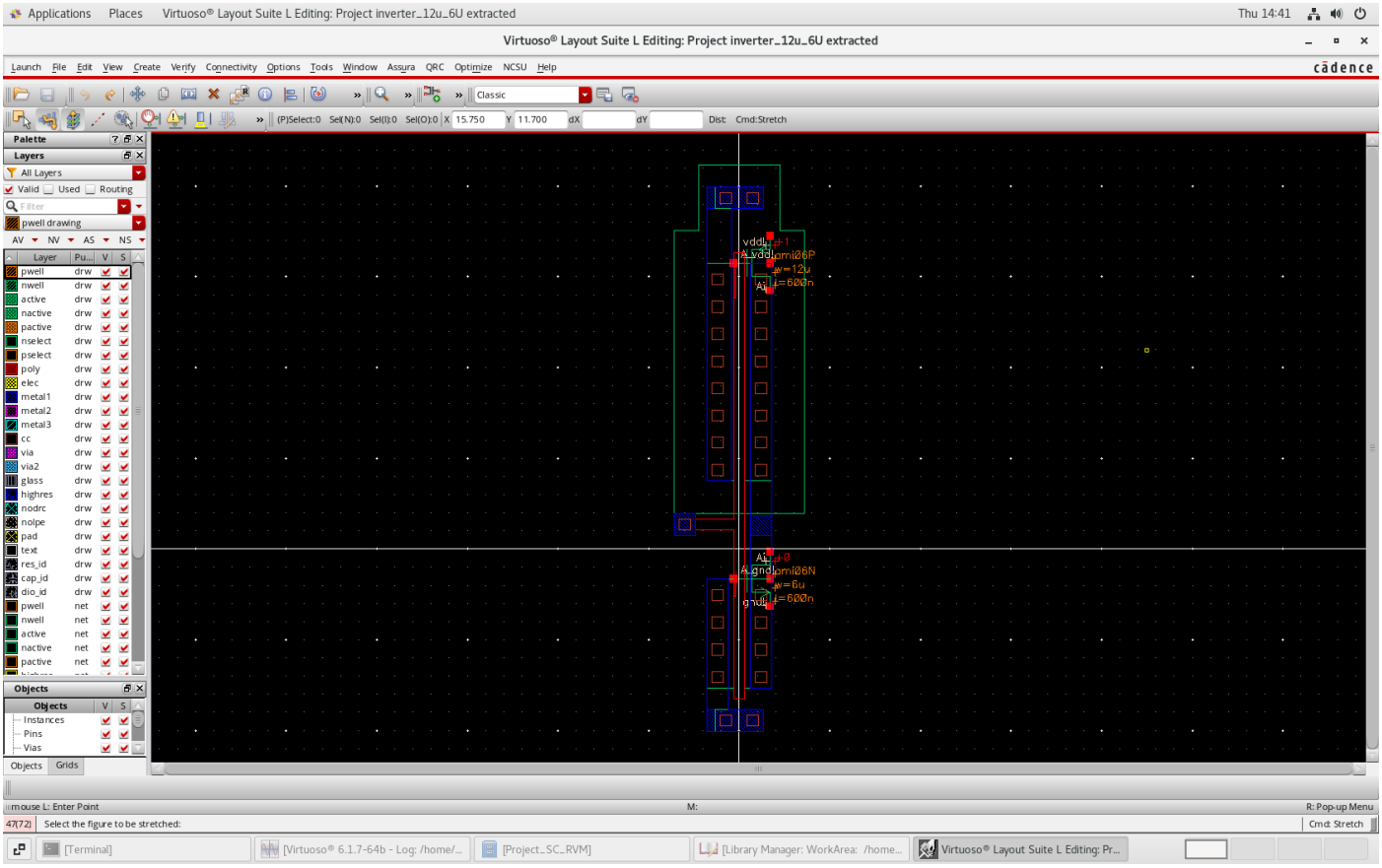
3.1.3 Layout



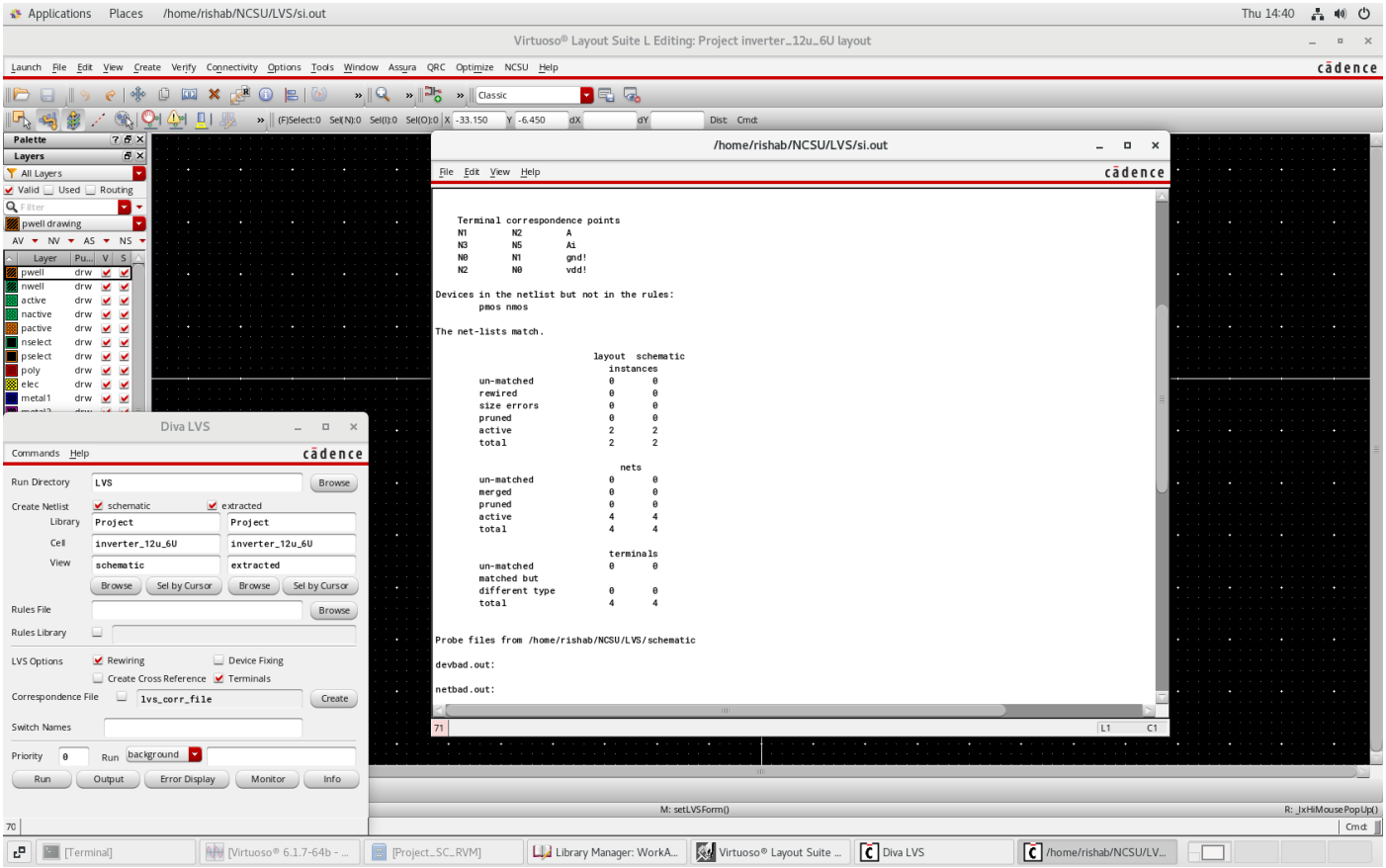
3.1.4 DRC



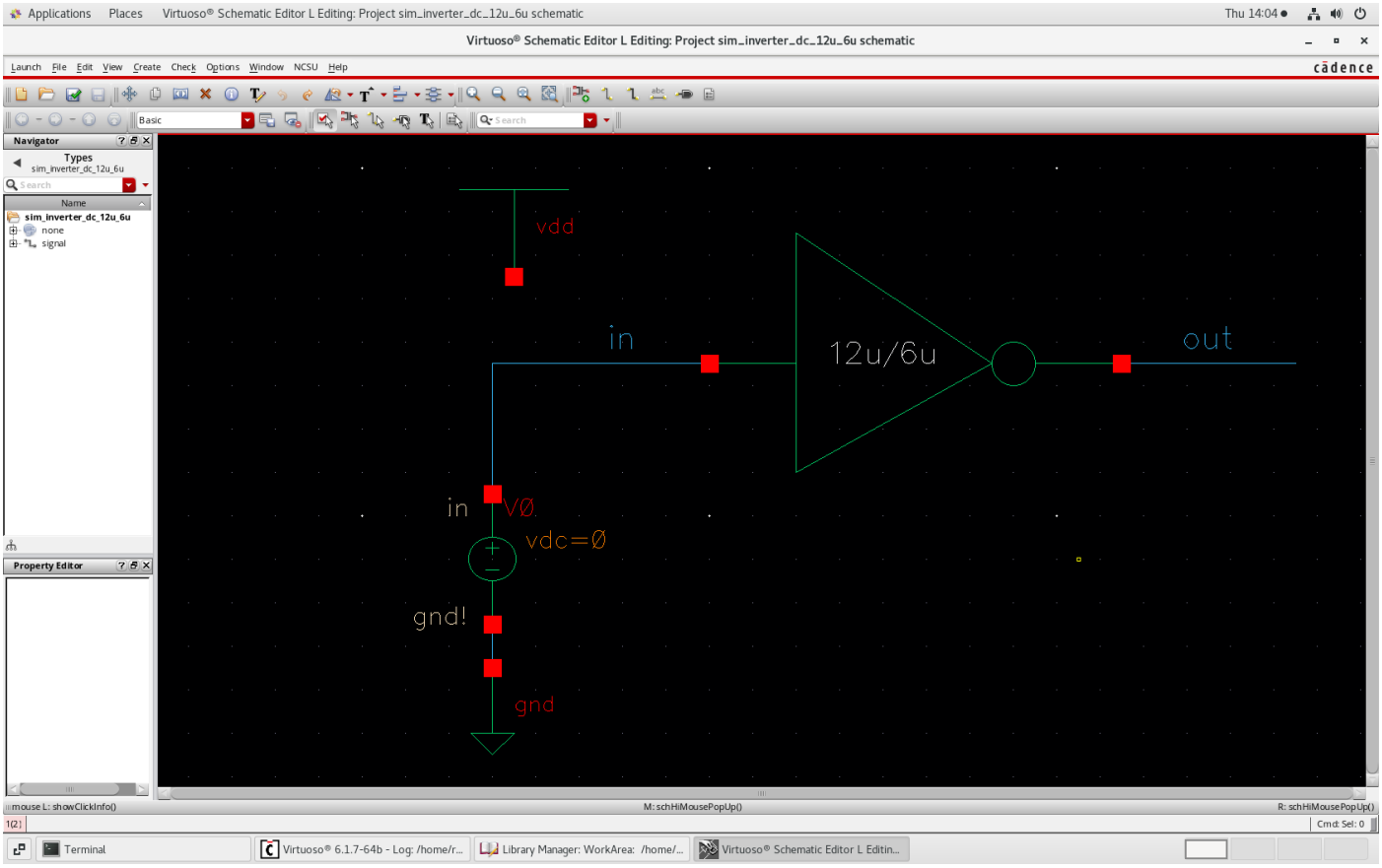
3.1.5 Extracted

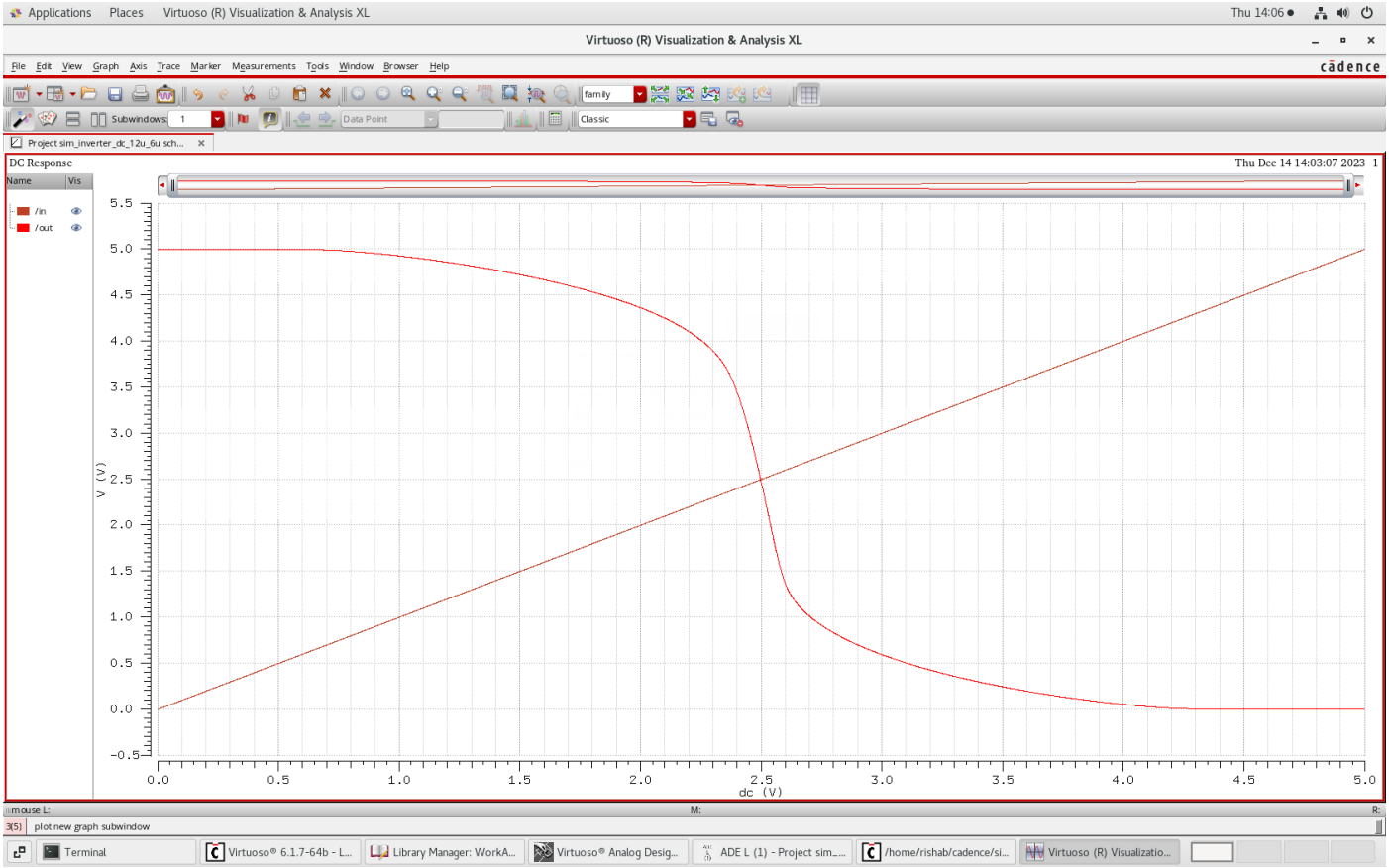


3.1.6 LVS



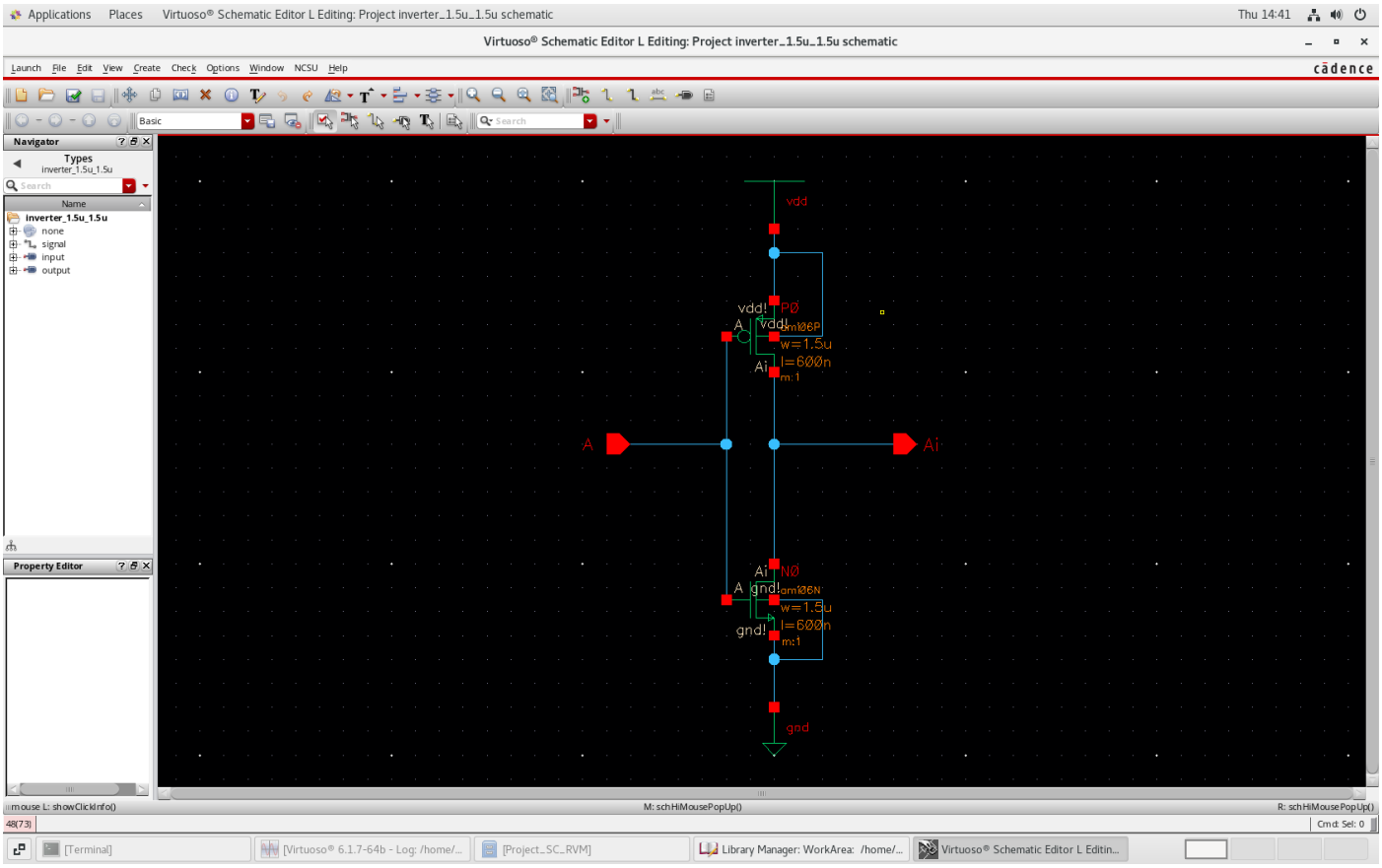
3.1.7 Simulation



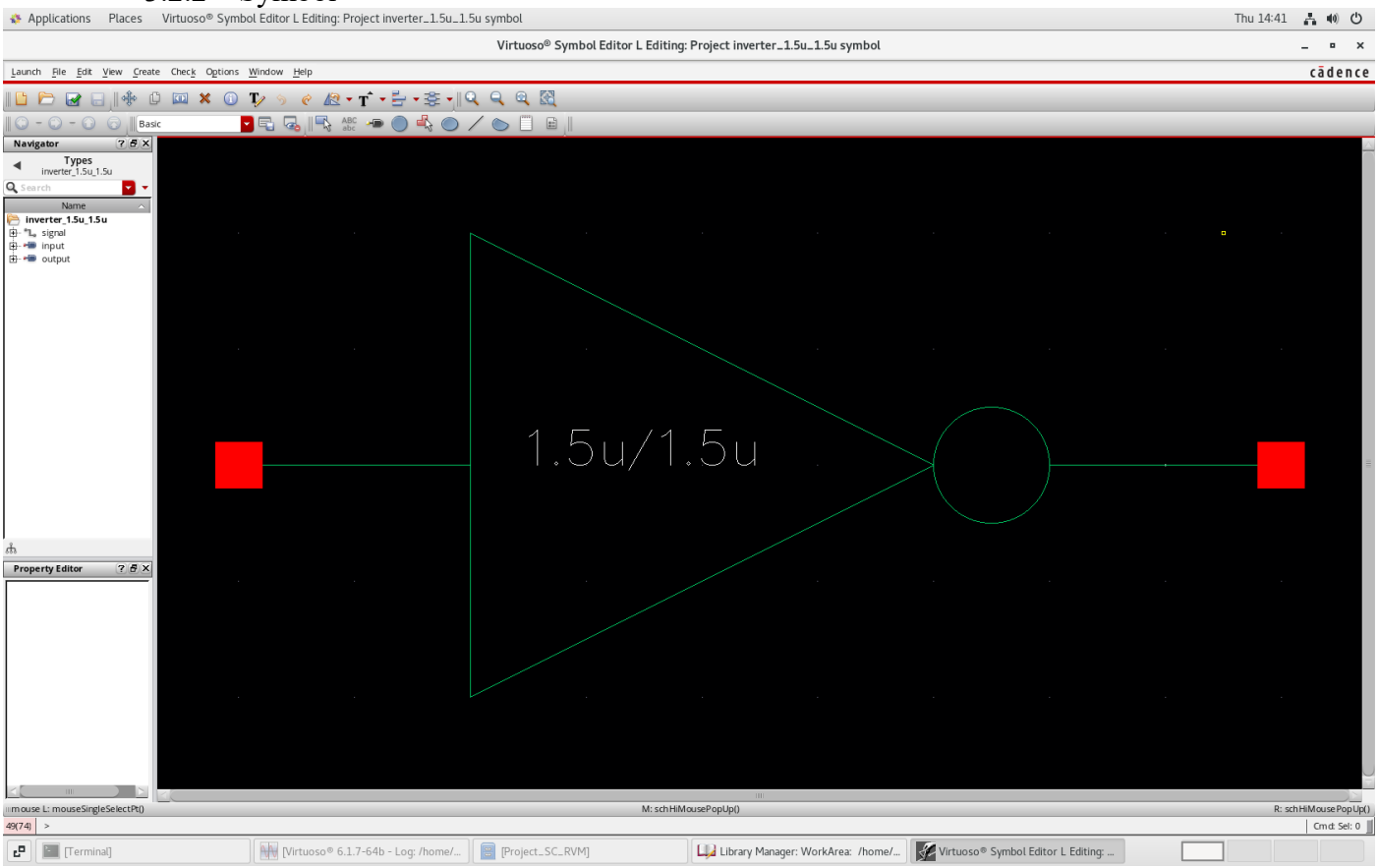


3.2 Inverter (1.5u/1.5u)

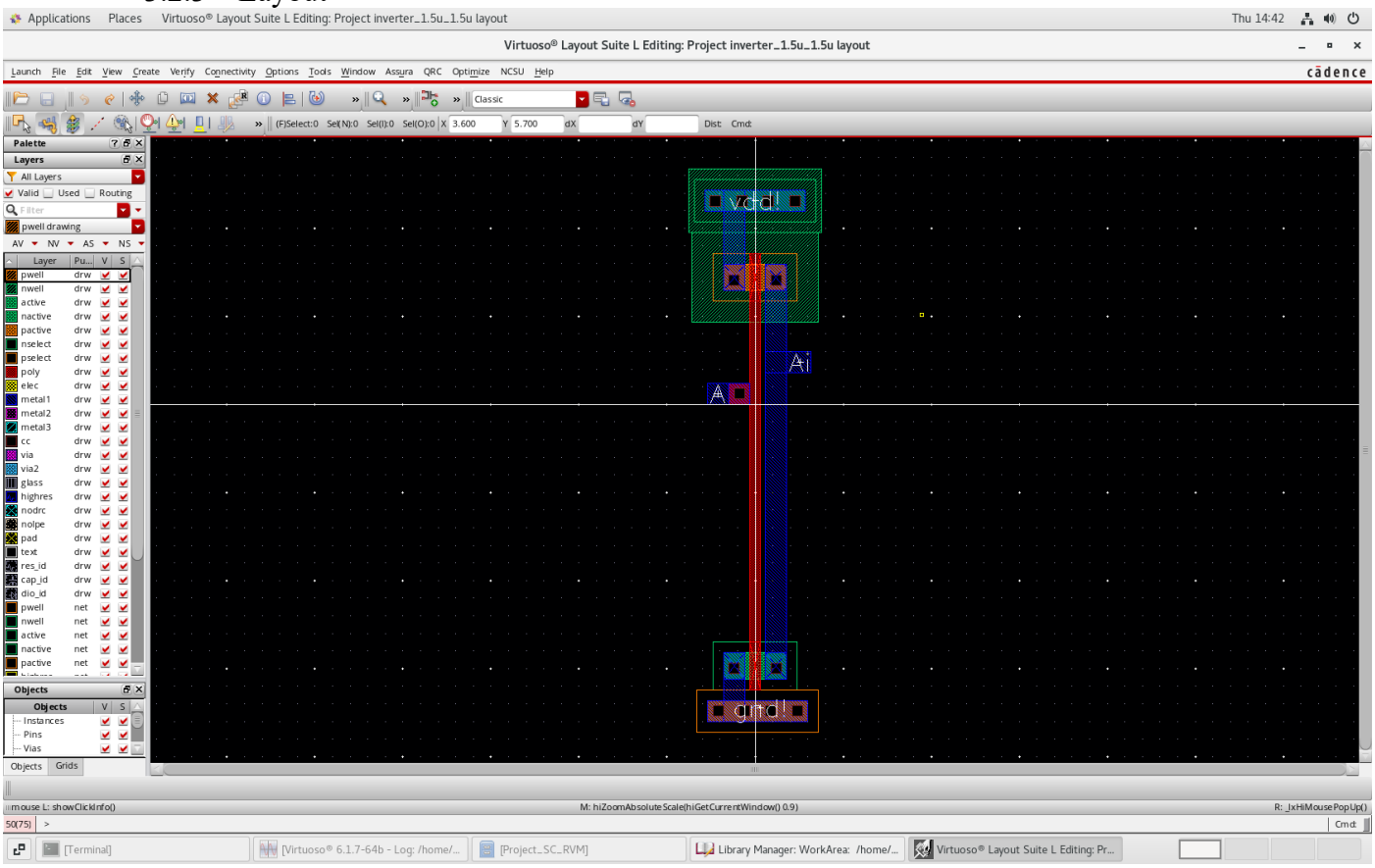
3.2.1 Schematic



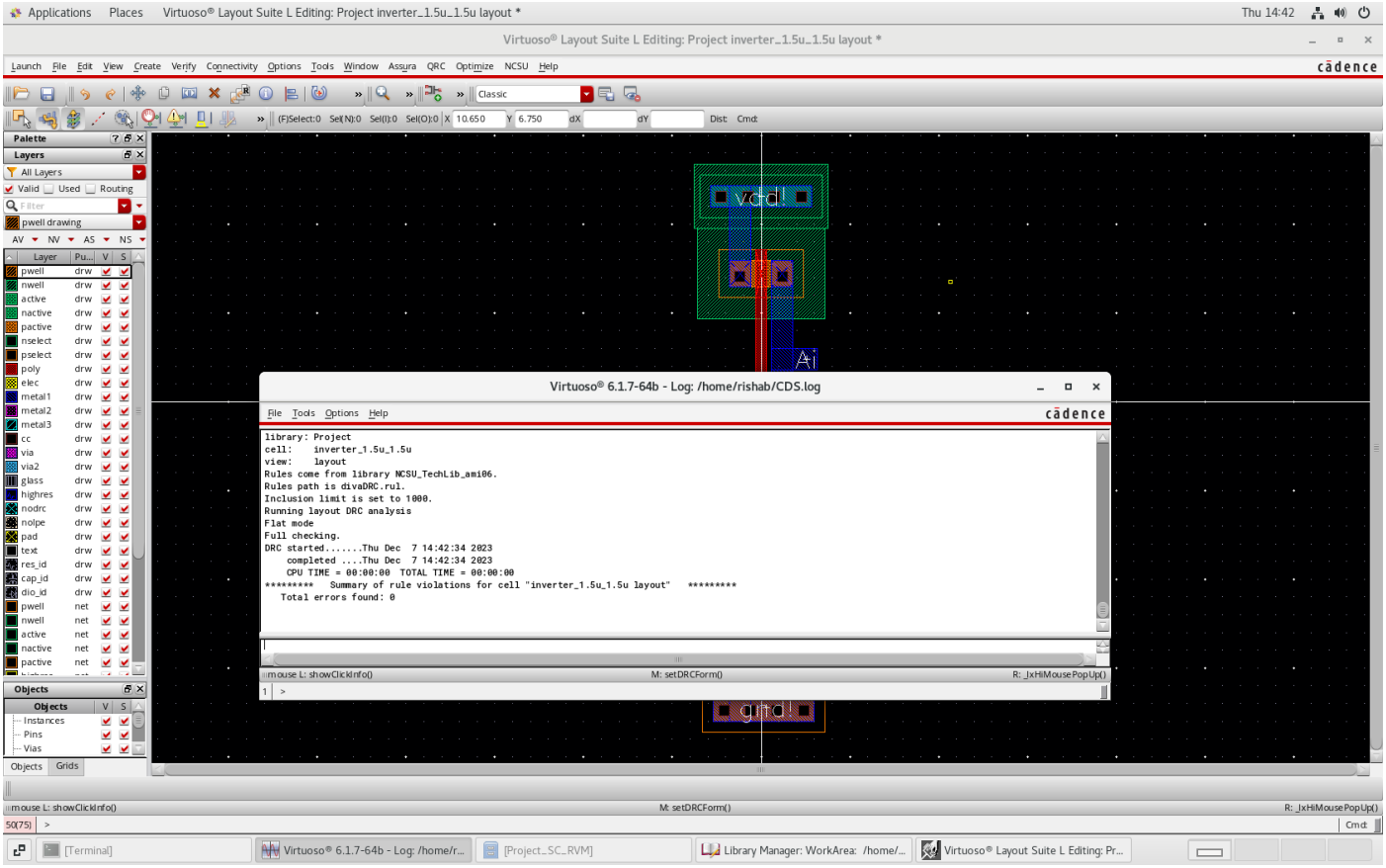
3.2.2 Symbol



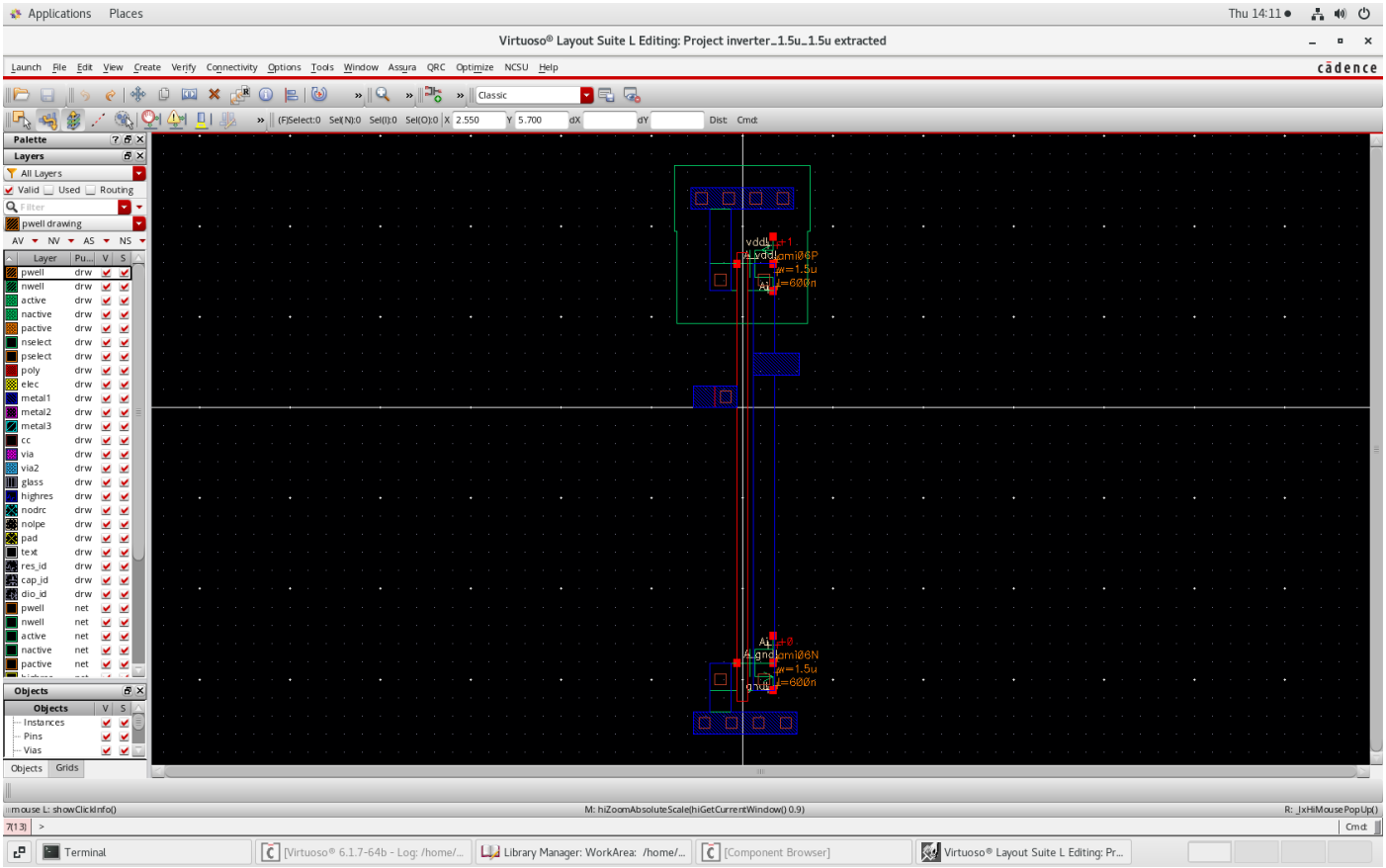
3.2.3 Layout



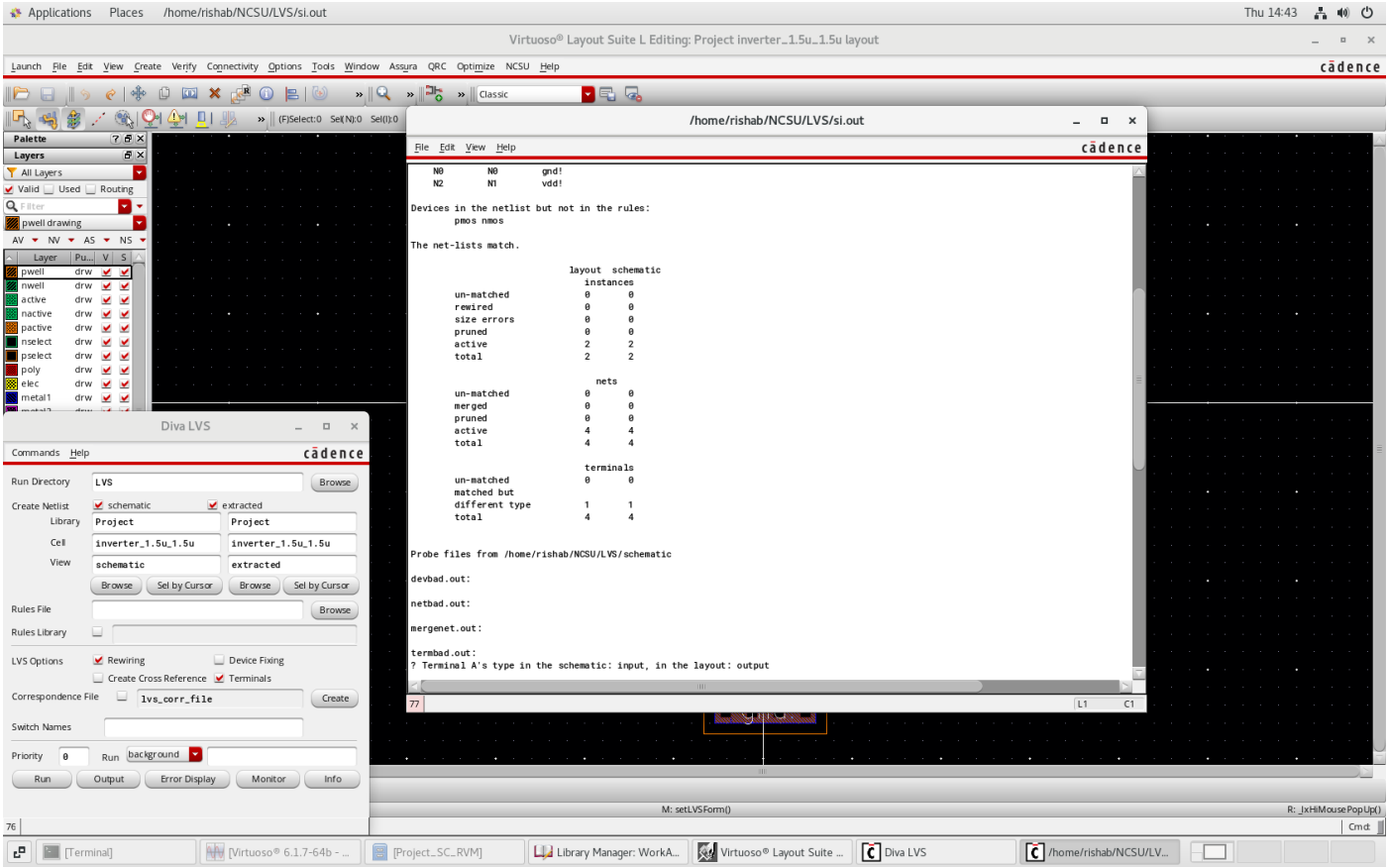
3.2.4 DRC



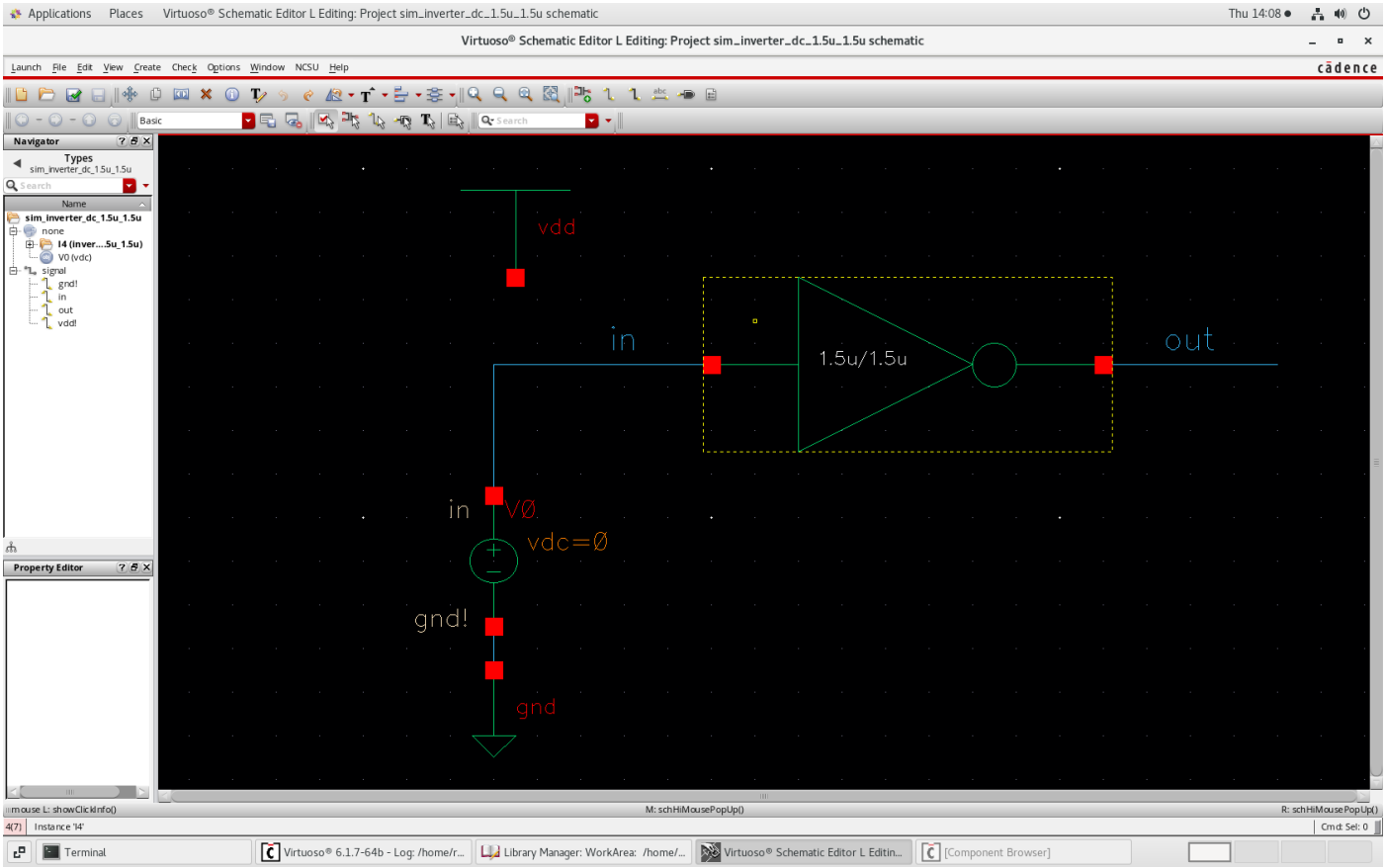
3.2.5 Extracted

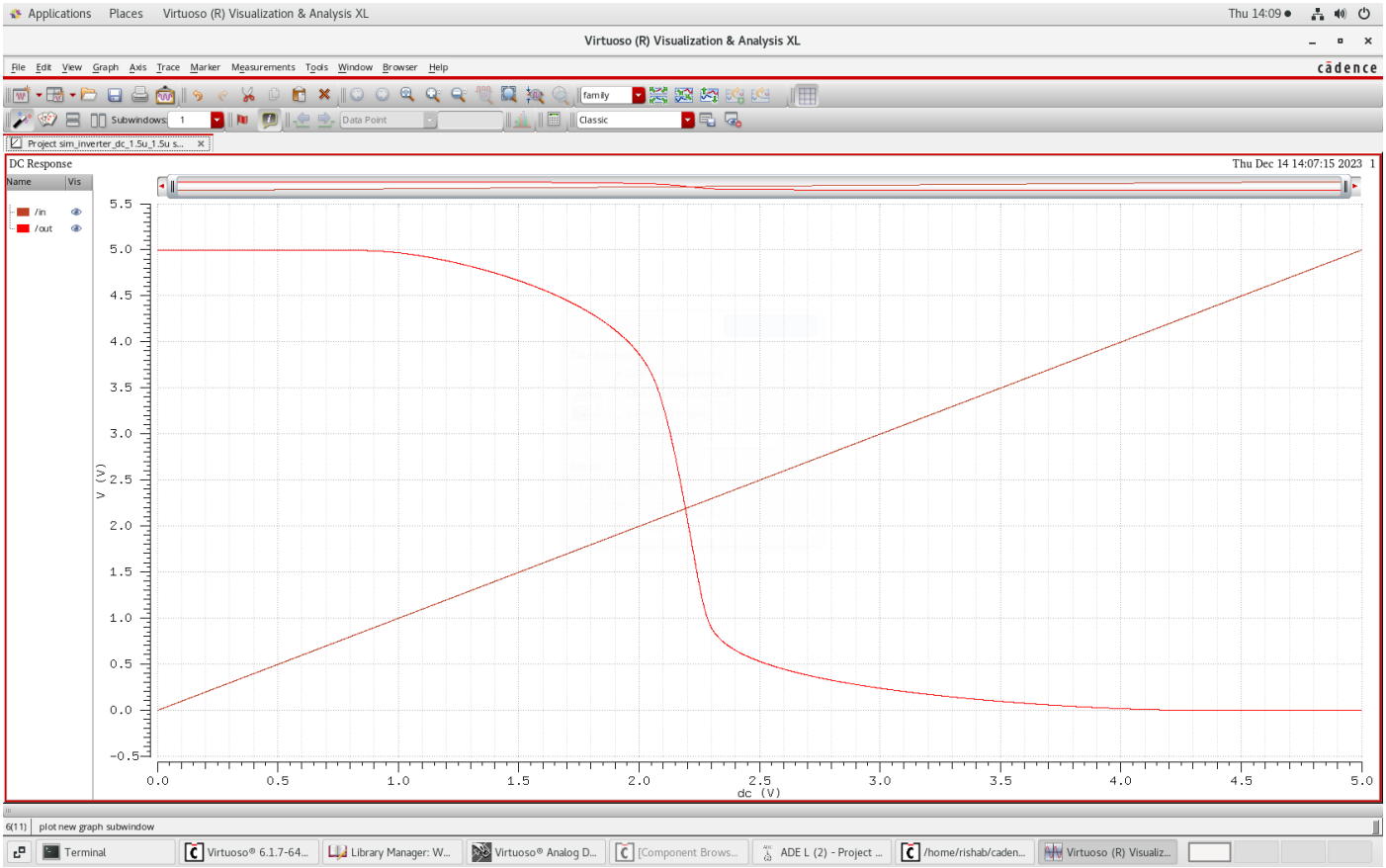


3.2.6 LVS



3.2.7 Simulation

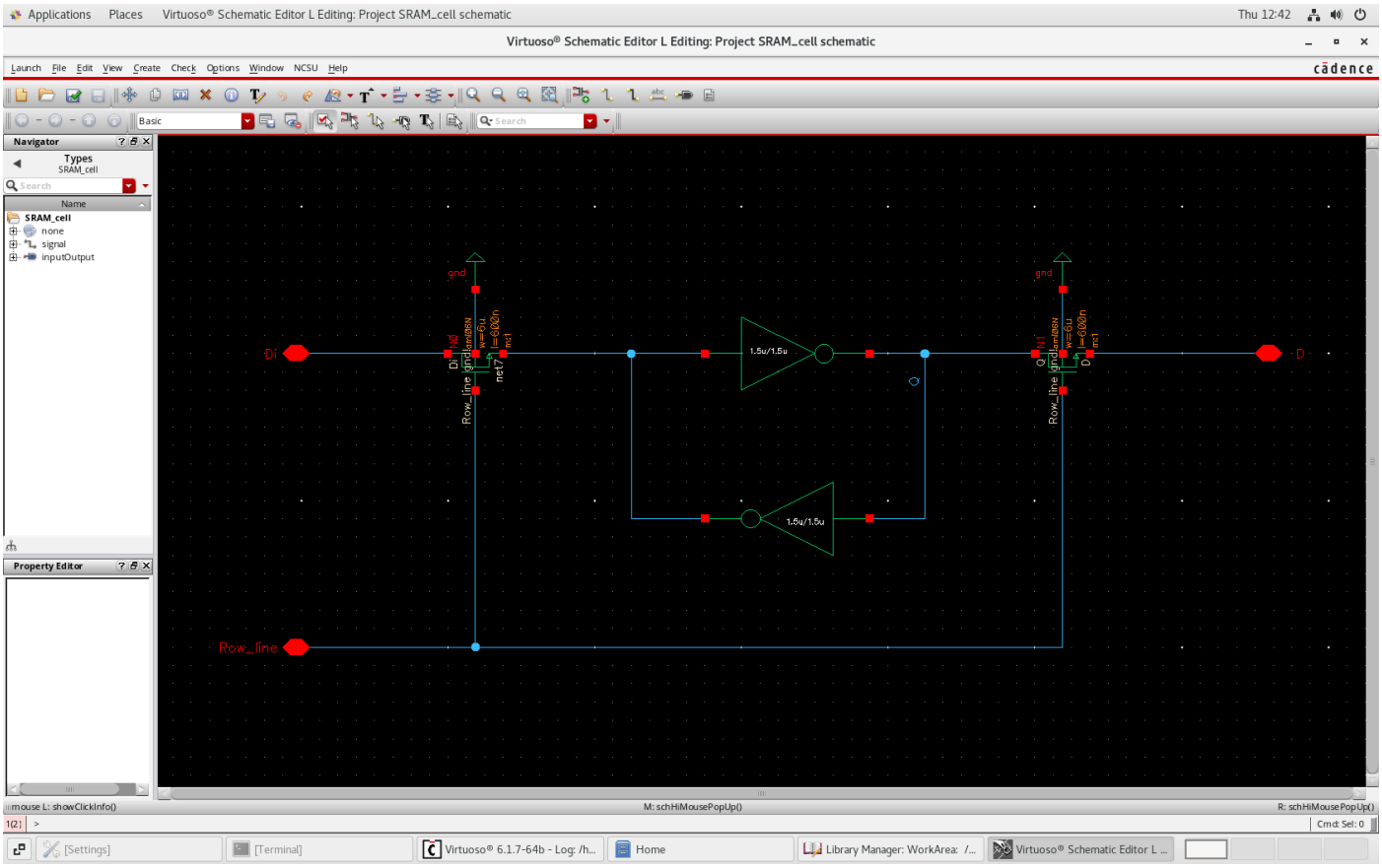




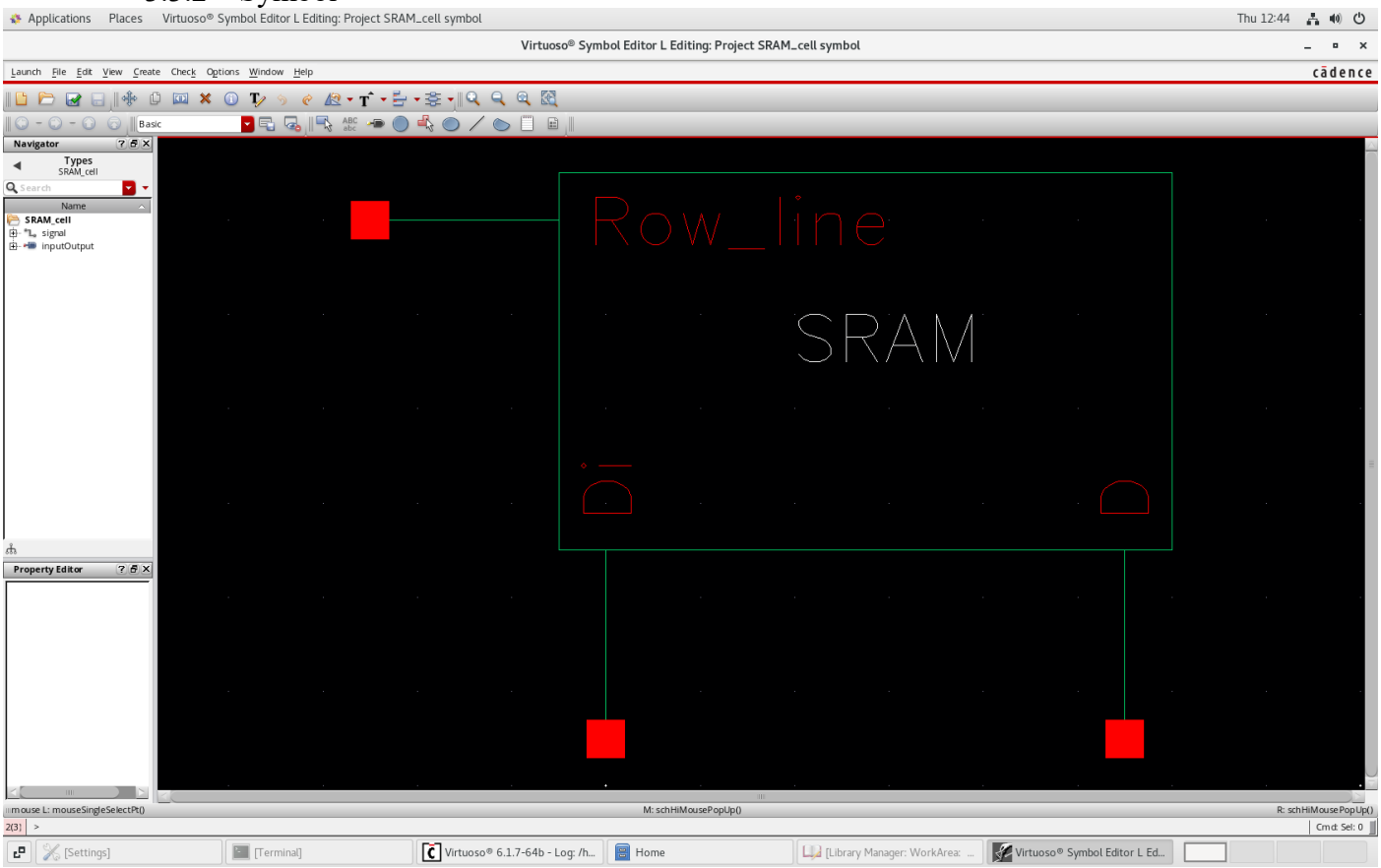
3.3 SRAM Cell

SRAM is a crucial element responsible for storing data and deciding where data is read and written. The SRAM is made up of two NMOS devices and two 1.5u/1.5u inverters. These inverters have a narrow width, making it easy to flip the cell. The inputs include D (data), Di (inverted data), and Row_line. When Row_line is in a high state, data is written, and when Row_line is low, data writing is halted.

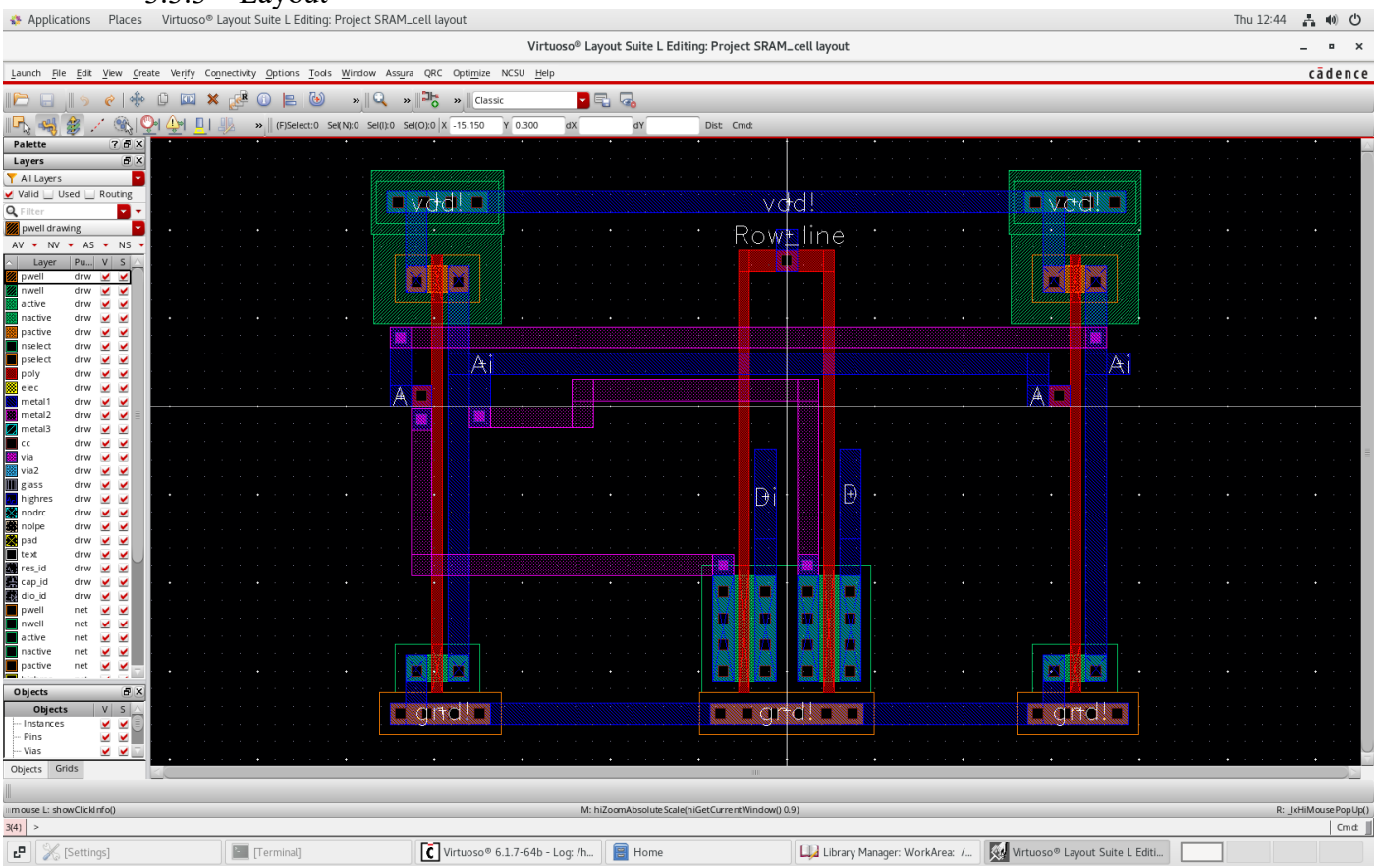
3.3.1 Schematic



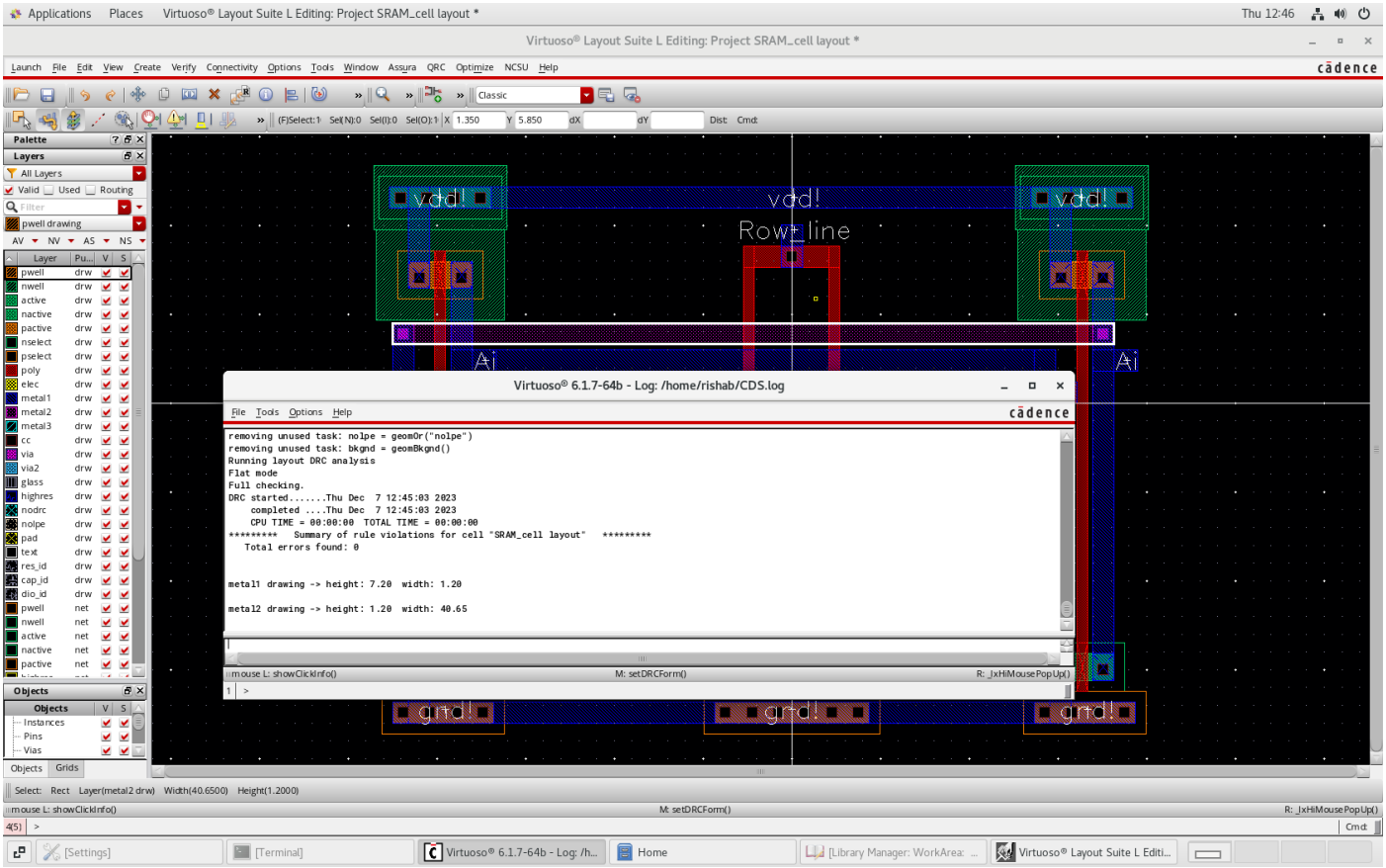
3.3.2 Symbol



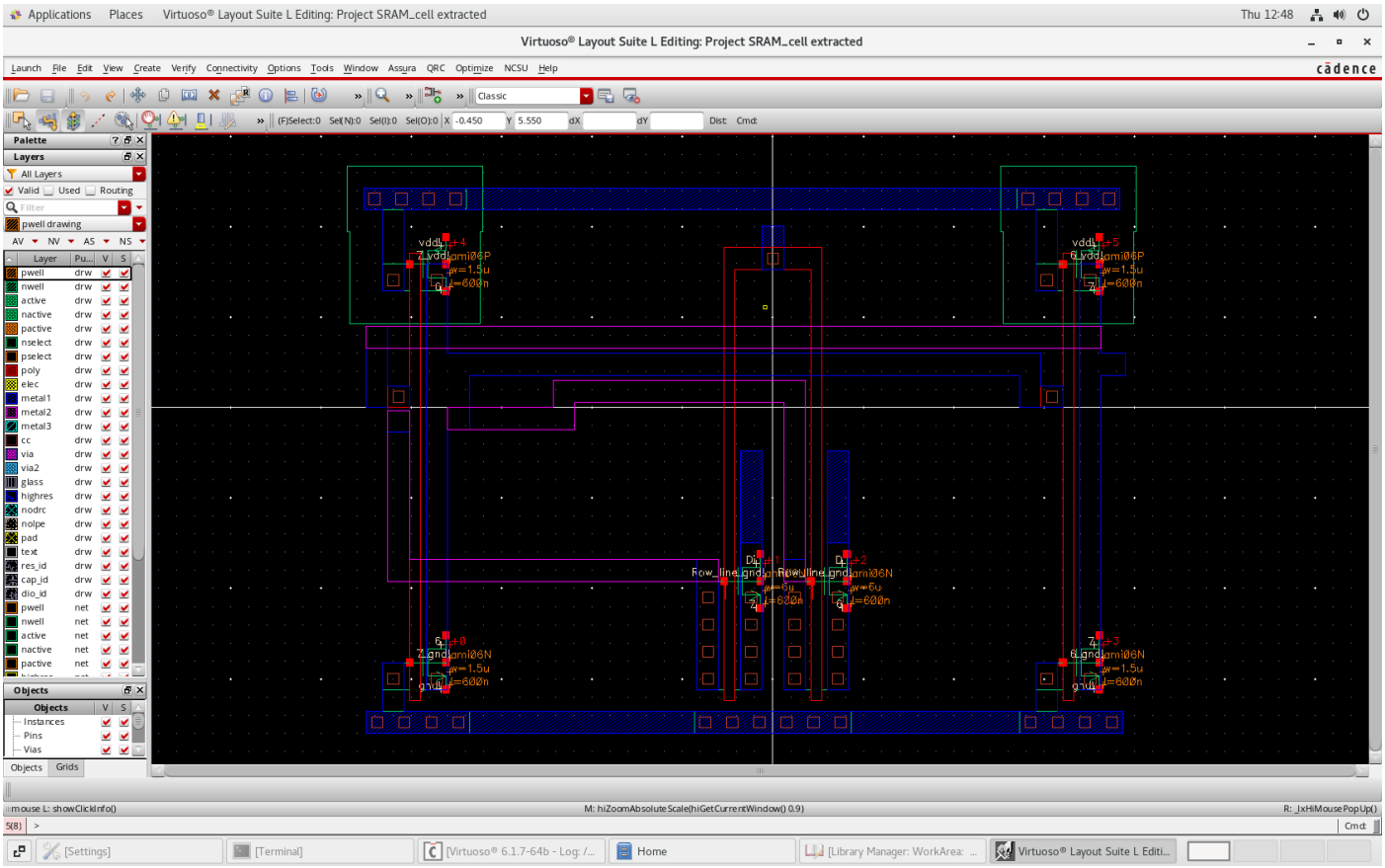
3.3.3 Layout



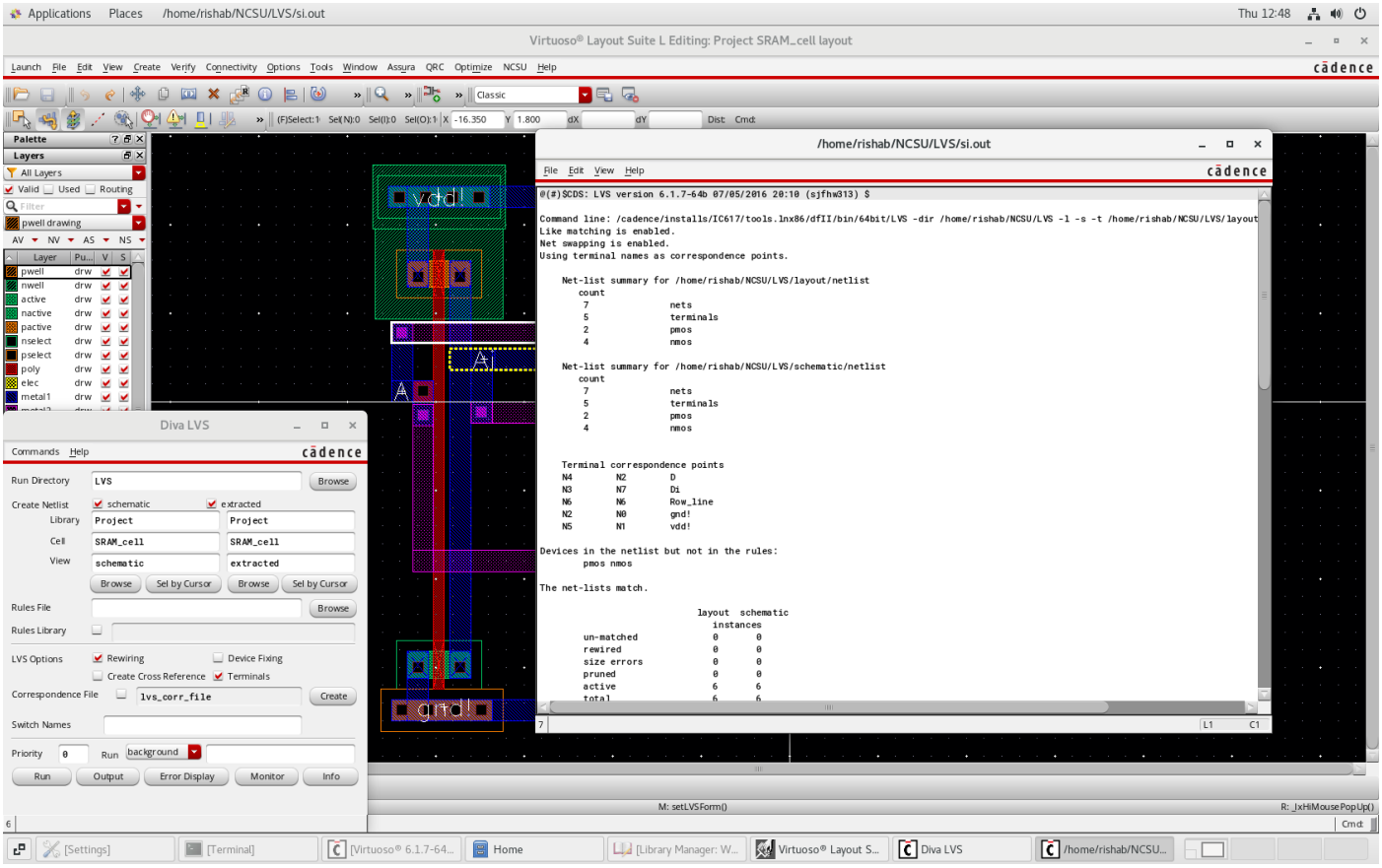
3.3.4 DRC



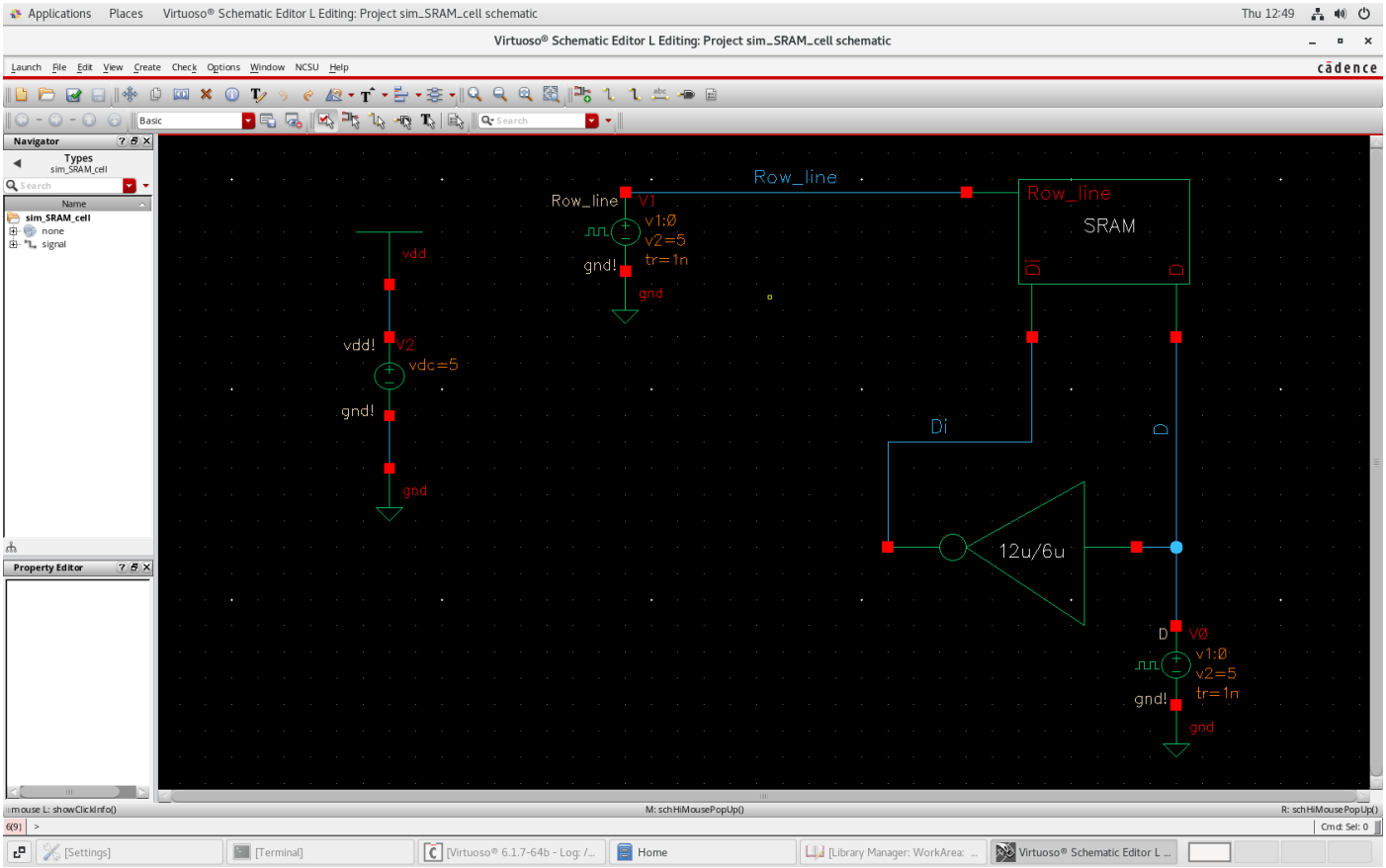
3.3.5 Extracted



3.3.6 LVS



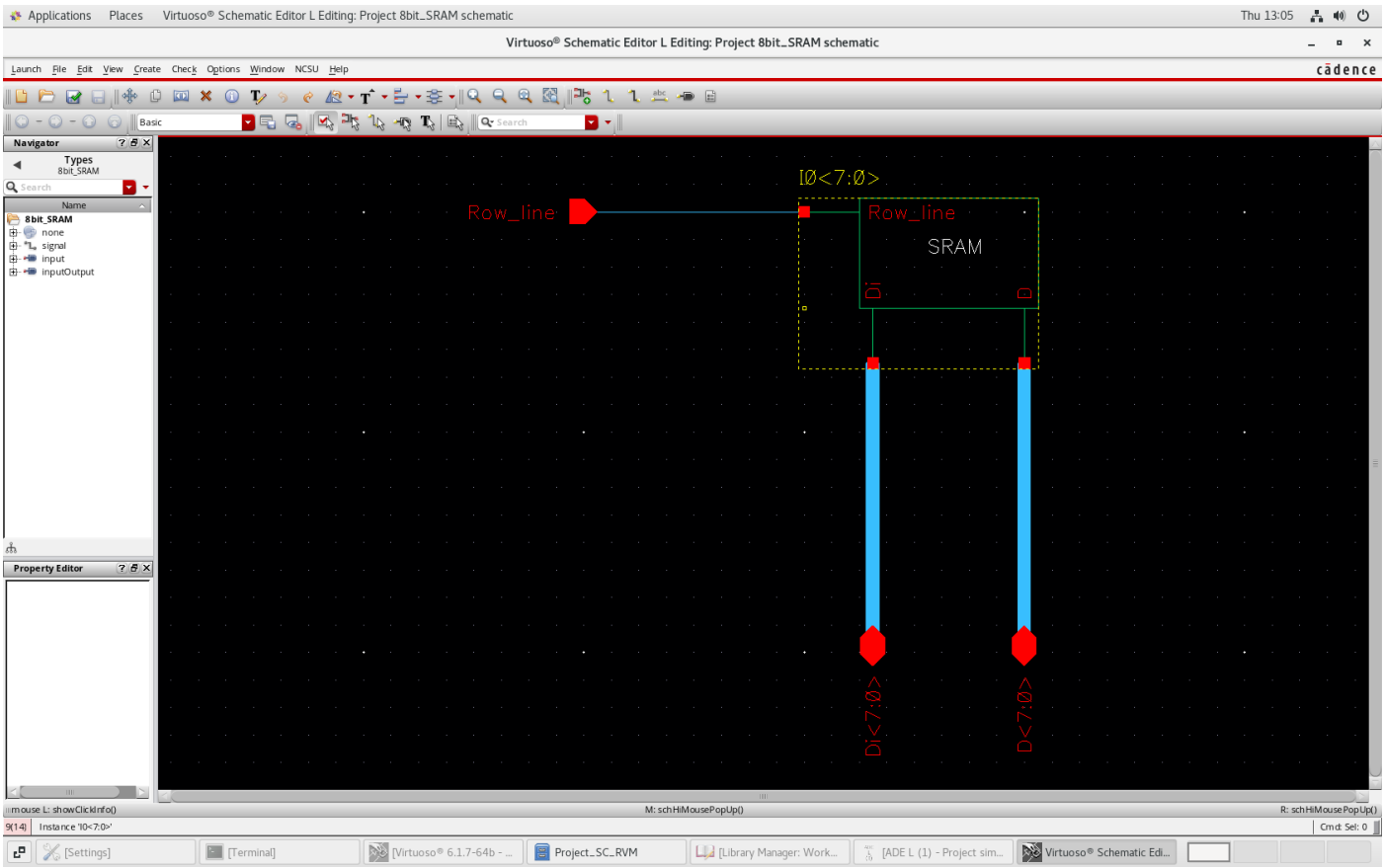
3.3.7 Simulation



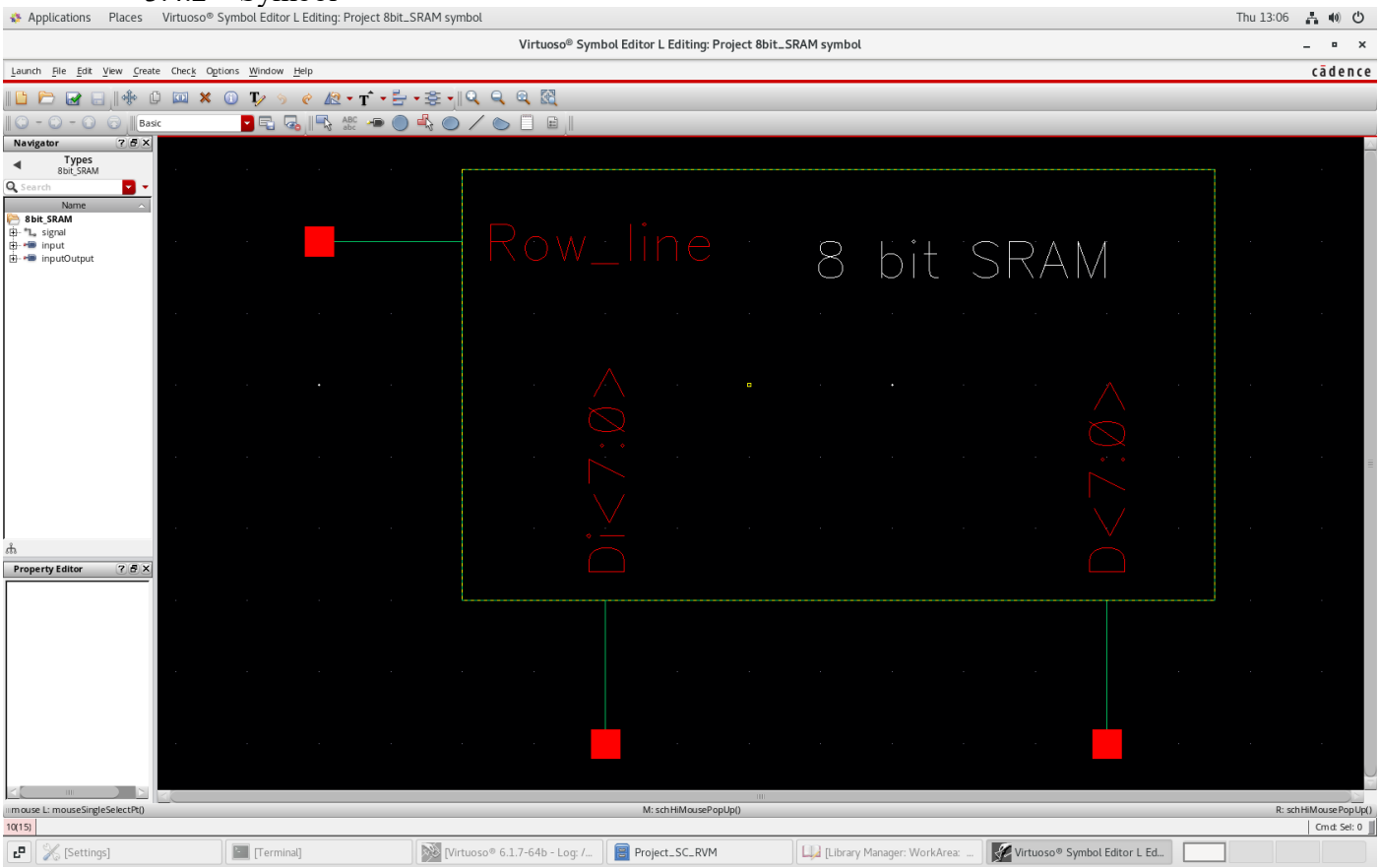
As observed, when Row_line is high, the current value of D gets stored and is also presented as the output. Conversely, when Row_line is low, the preceding value of D is retained and consistently serves as the output.

3.4 8-bit SRAM

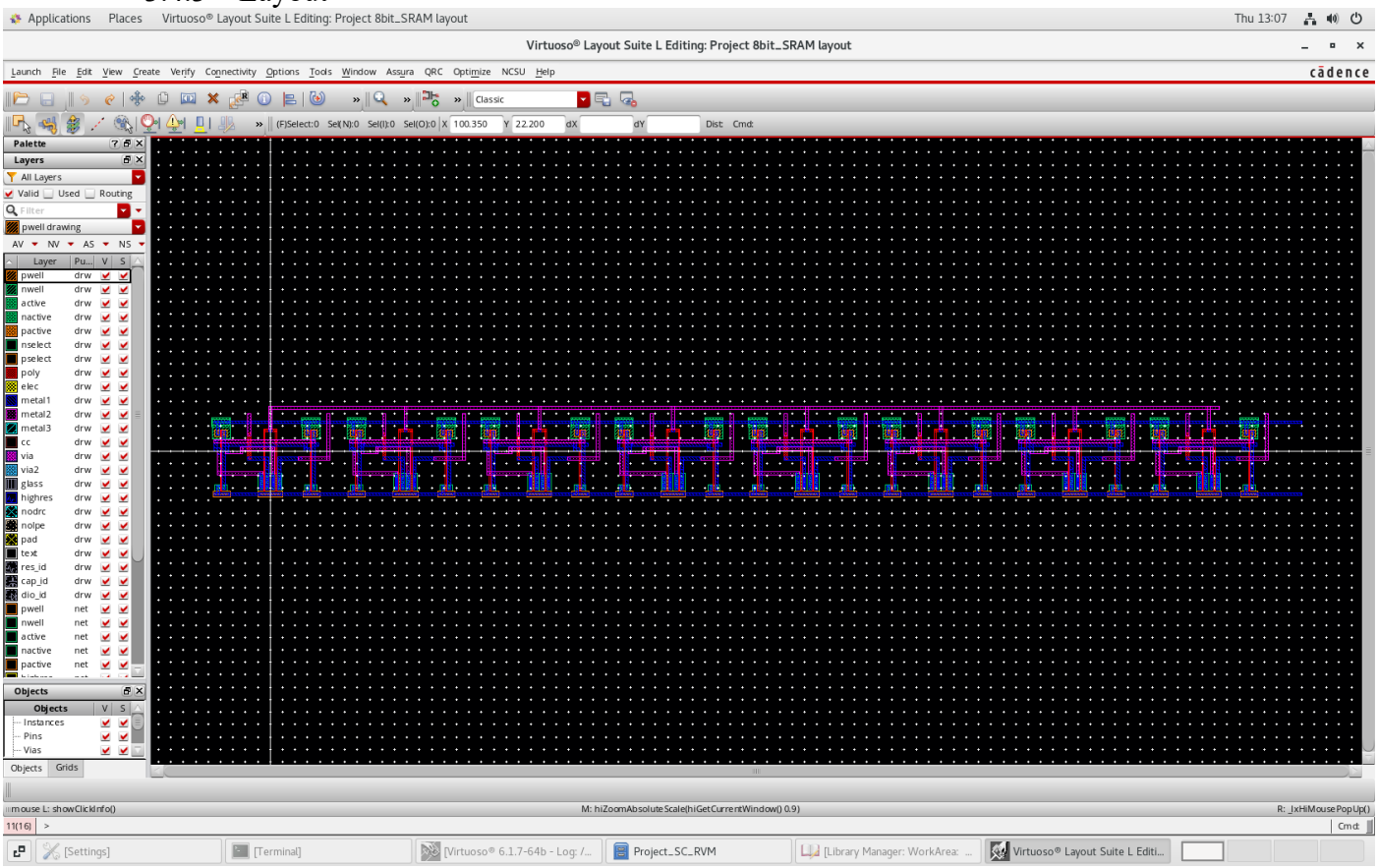
3.4.1 Schematic



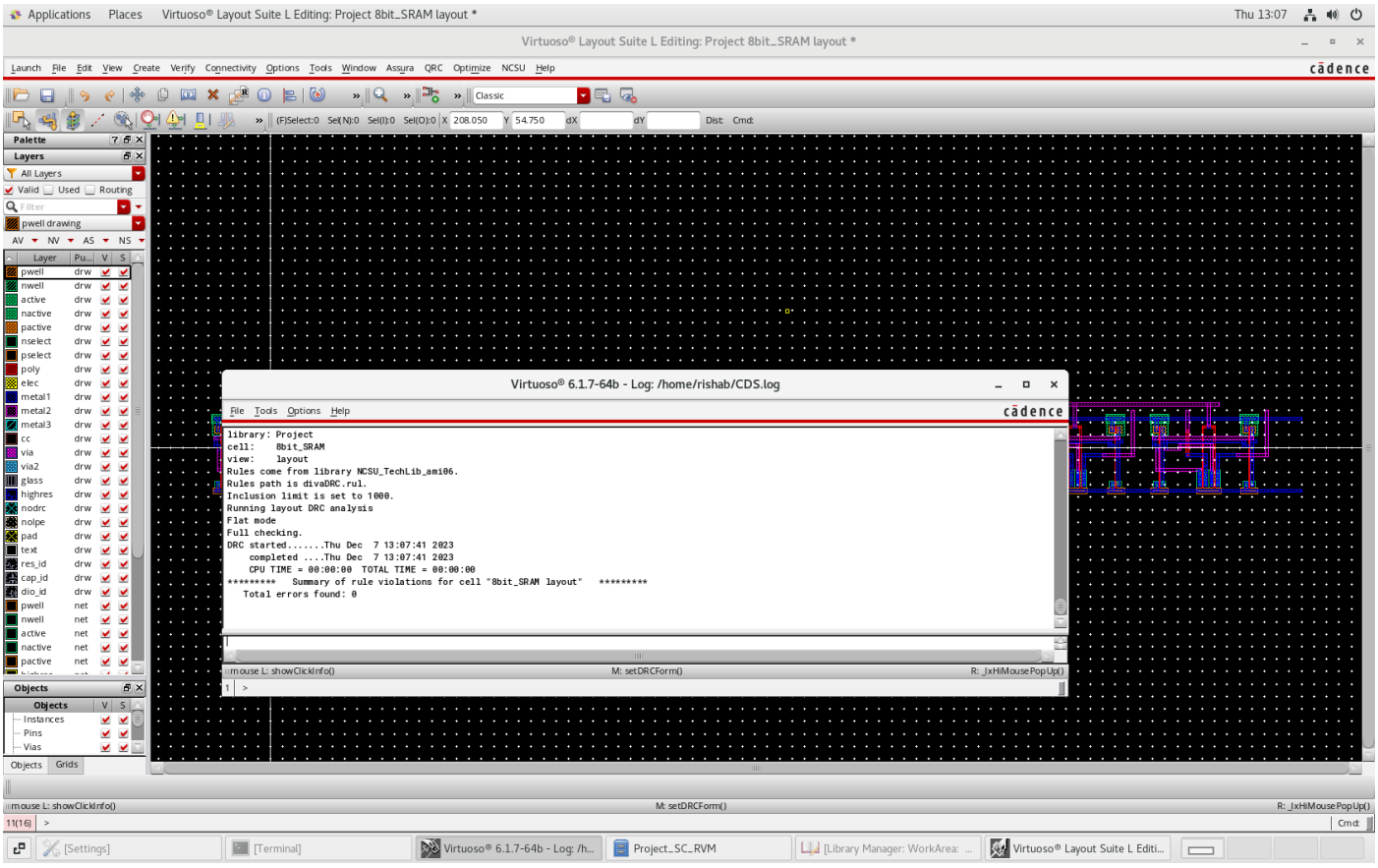
3.4.2 Symbol



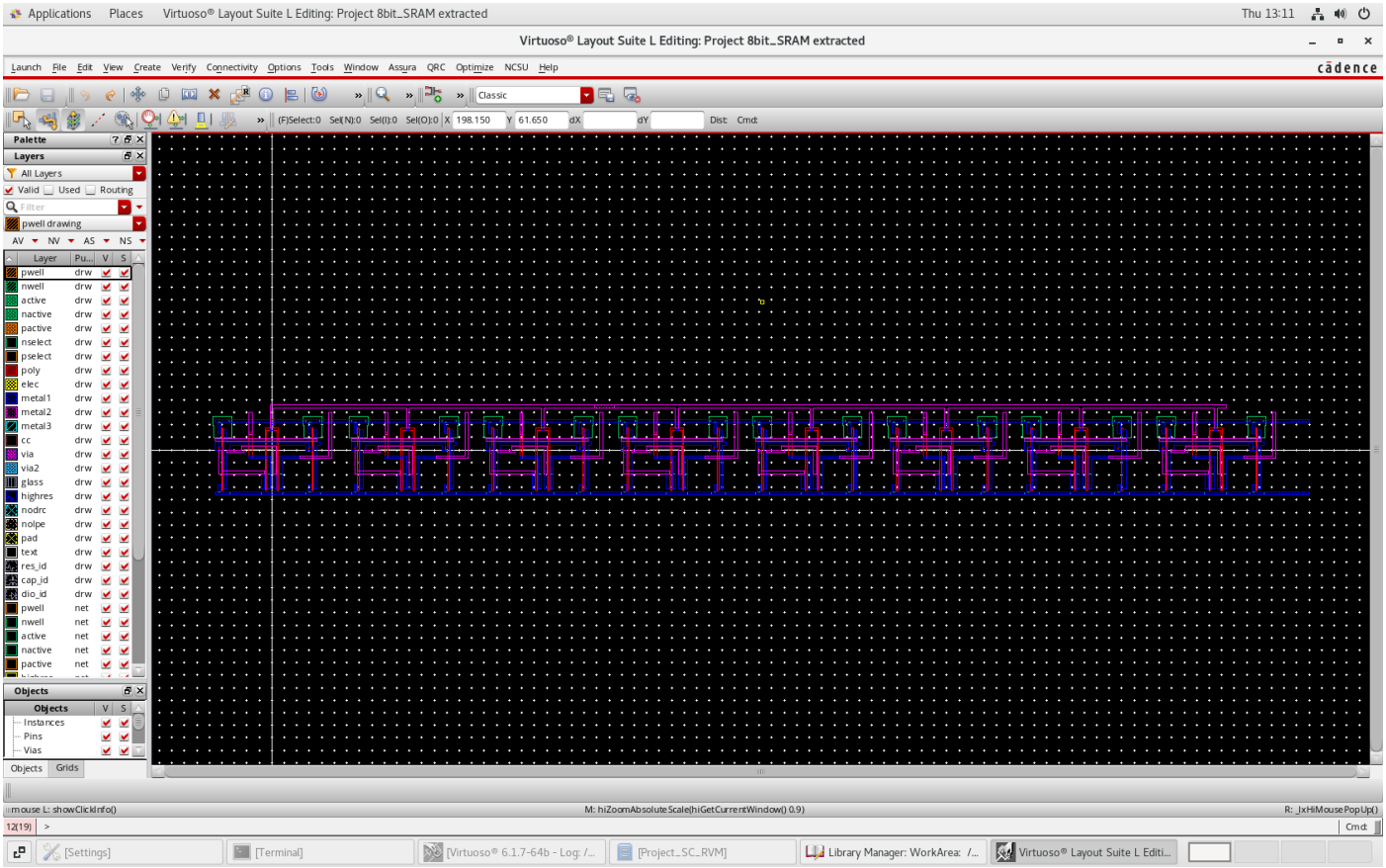
3.4.3 Layout



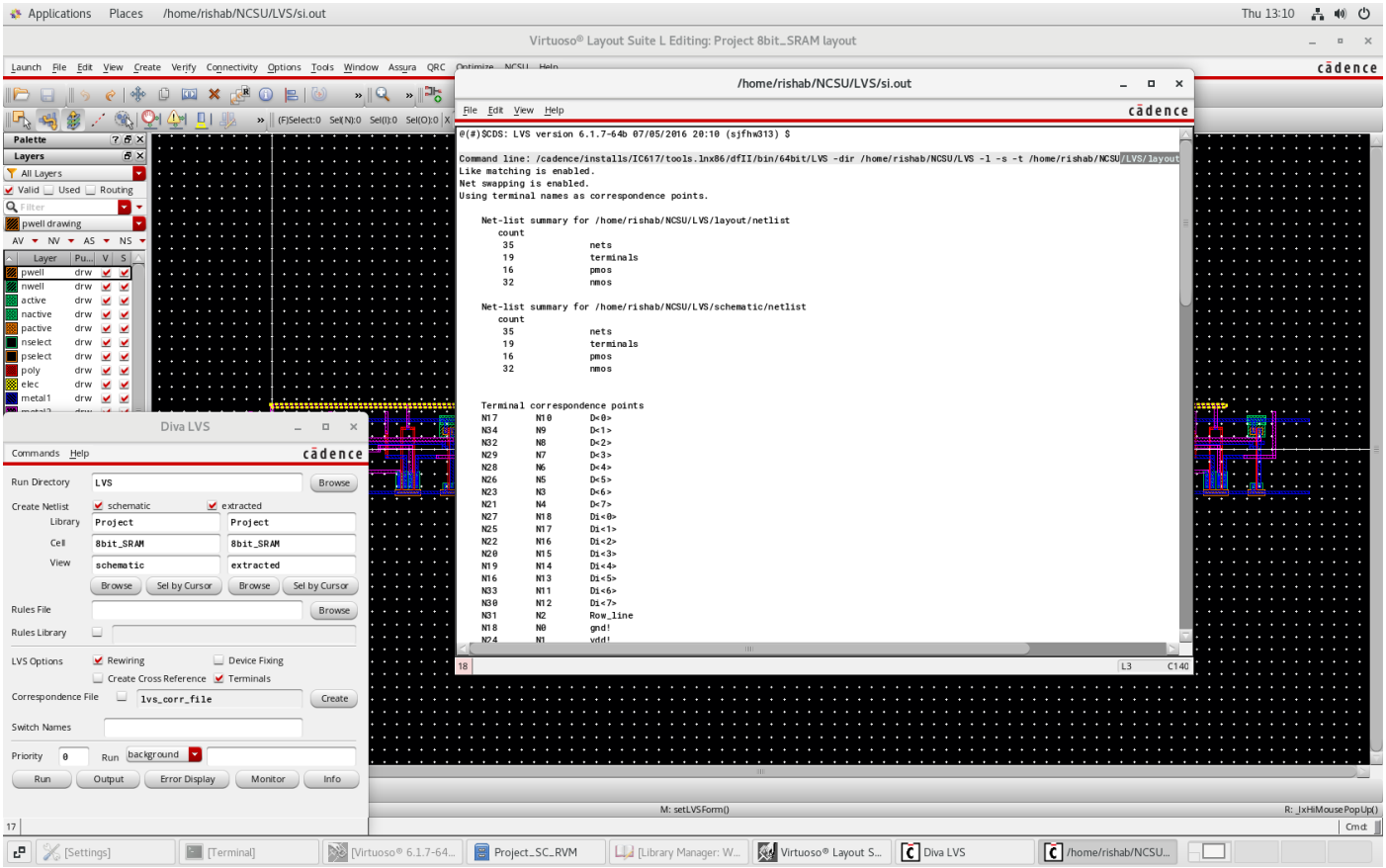
3.4.4 DRC



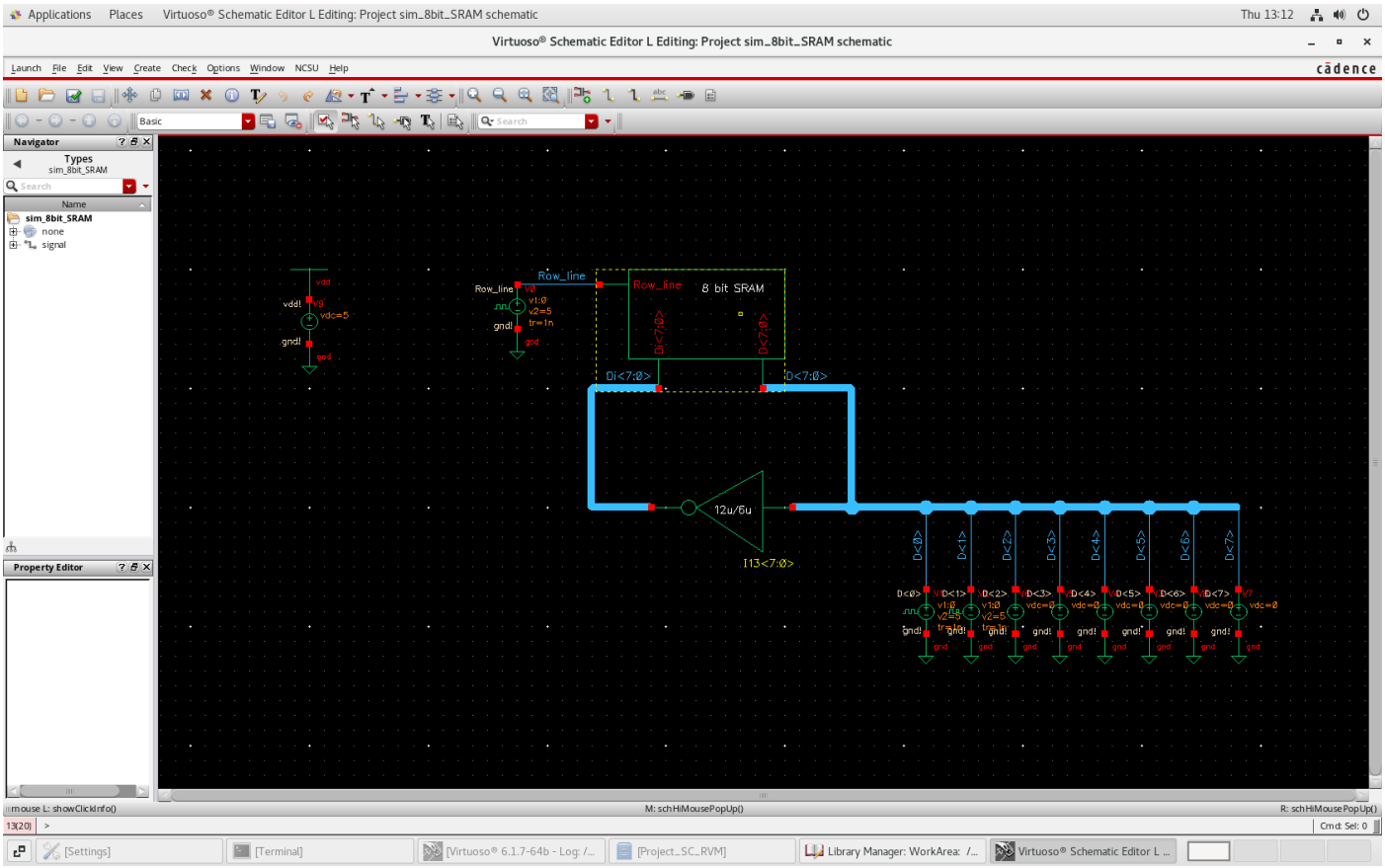
3.4.5 Extracted



3.4.6 LVS



3.4.7 Simulation

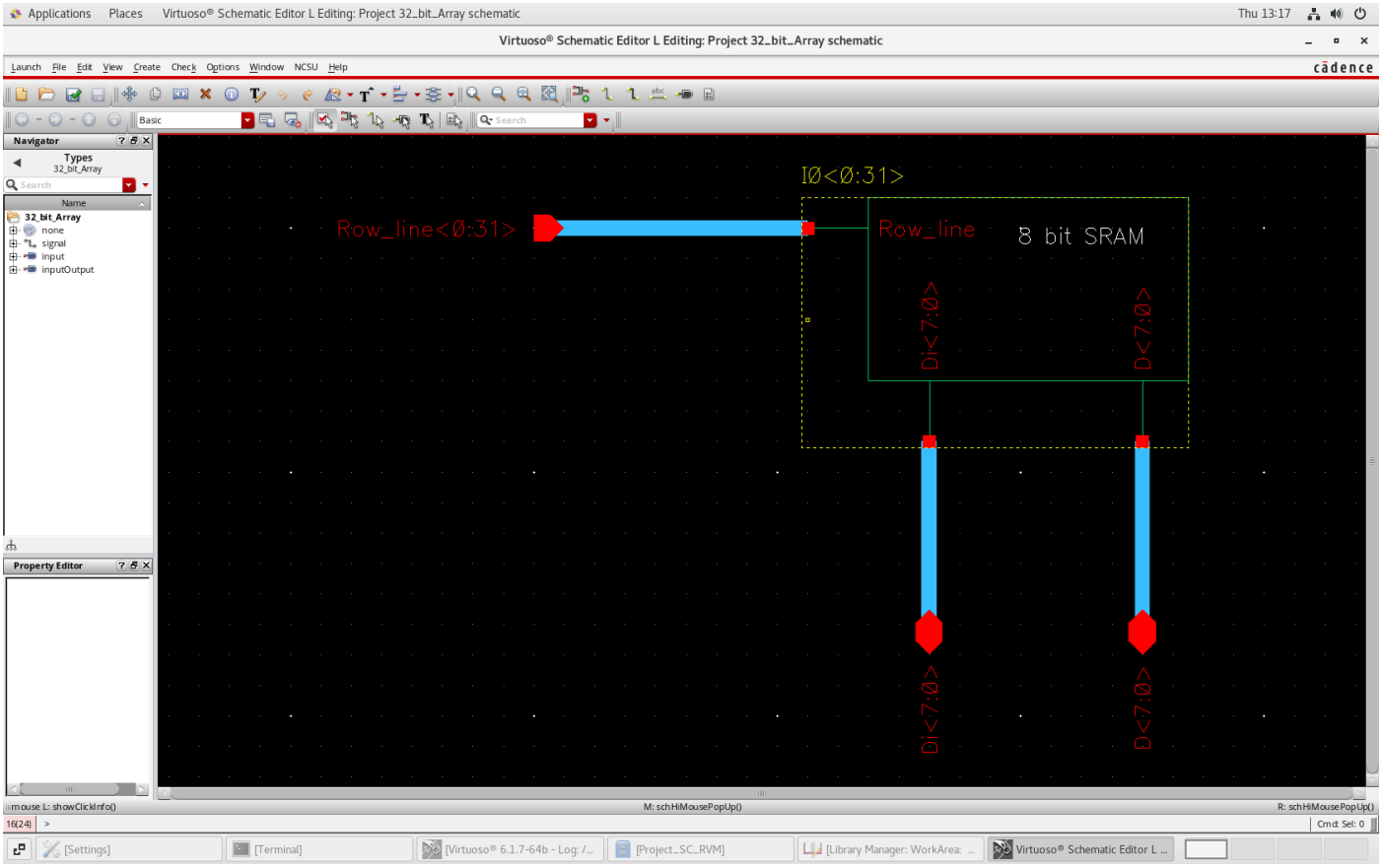


The behavior of the 8-bit SRAM mirrors that of the 1-bit SRAM. When Row_line is high, I0<0> mirrors D<0>, and I0<1> mirrors D<1>. However, when Row_line transitions to a low state, the last values from D<0> and D<1> are retained, serving as the output for I0<0> and I0<1>.

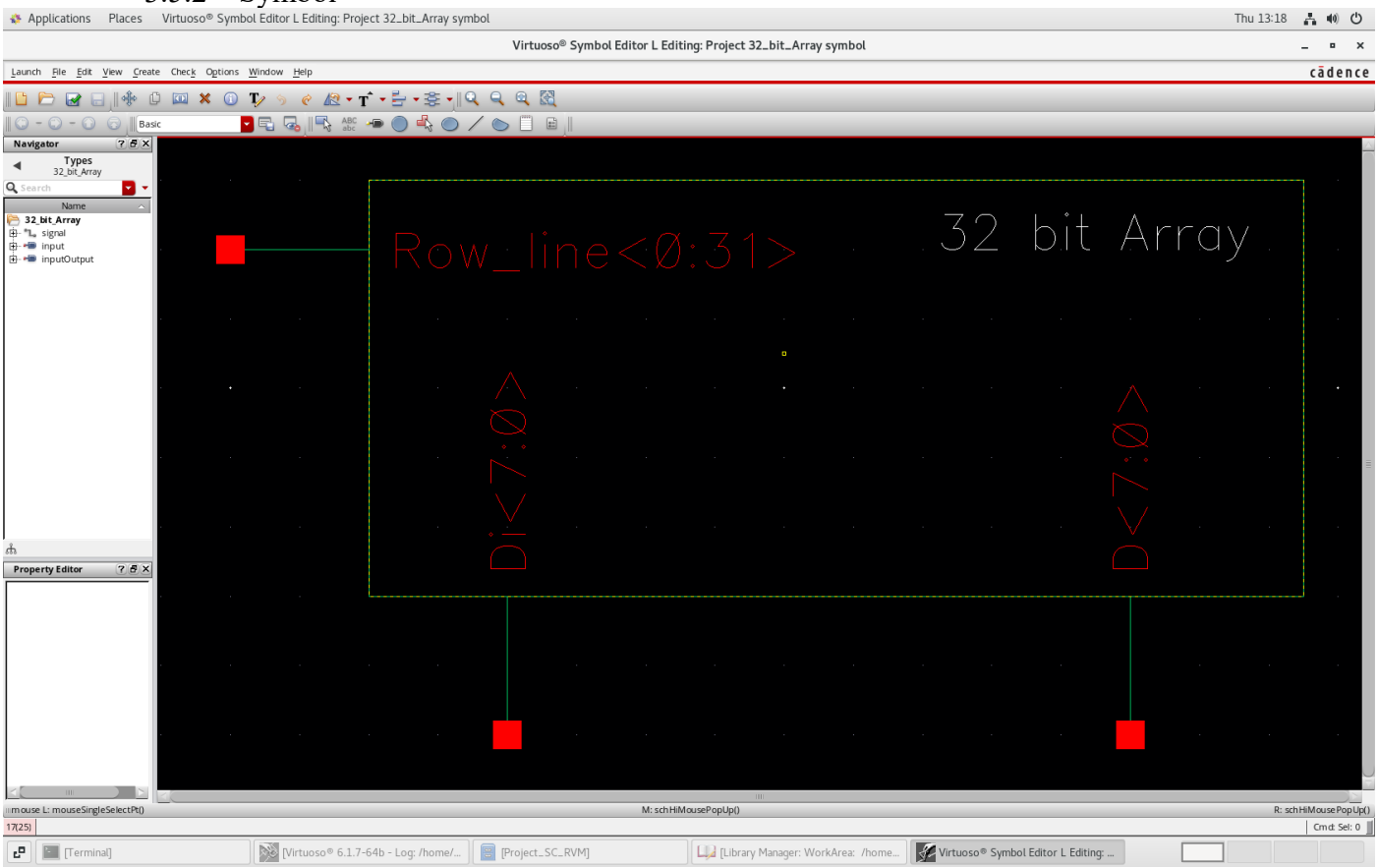
3.5 32-bit Array

Next, we build the 32-bit array responsible for storing the memory for 32 words, each with 8 bits. To accomplish this, we use the 8-bit SRAM we designed earlier and instantiate it 32 times. This is achieved using an array. Since there are 32 individual 8-bit SRAMs, we also generate 32 Row_line using a bus.

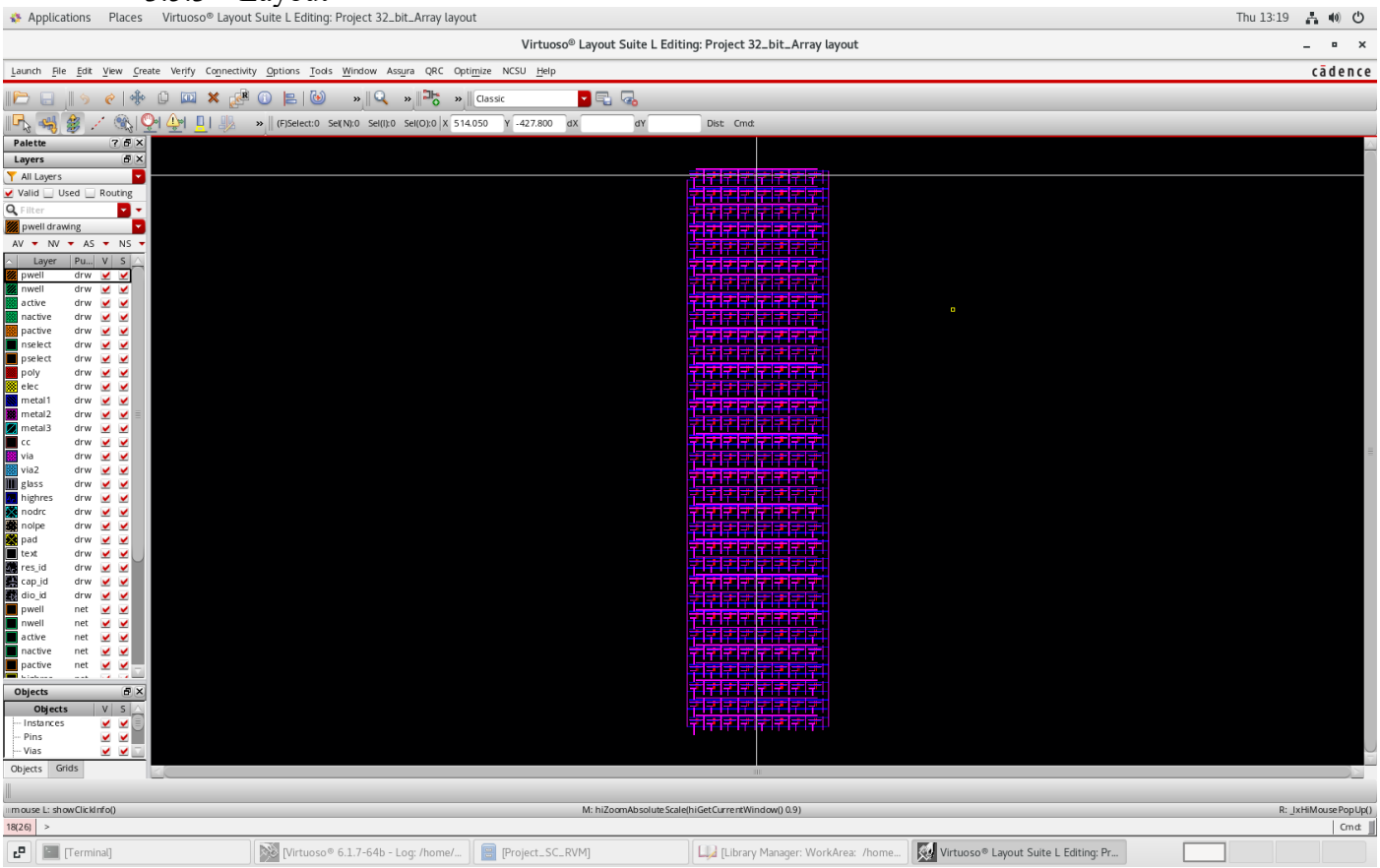
3.5.1 Schematic



3.5.2 Symbol

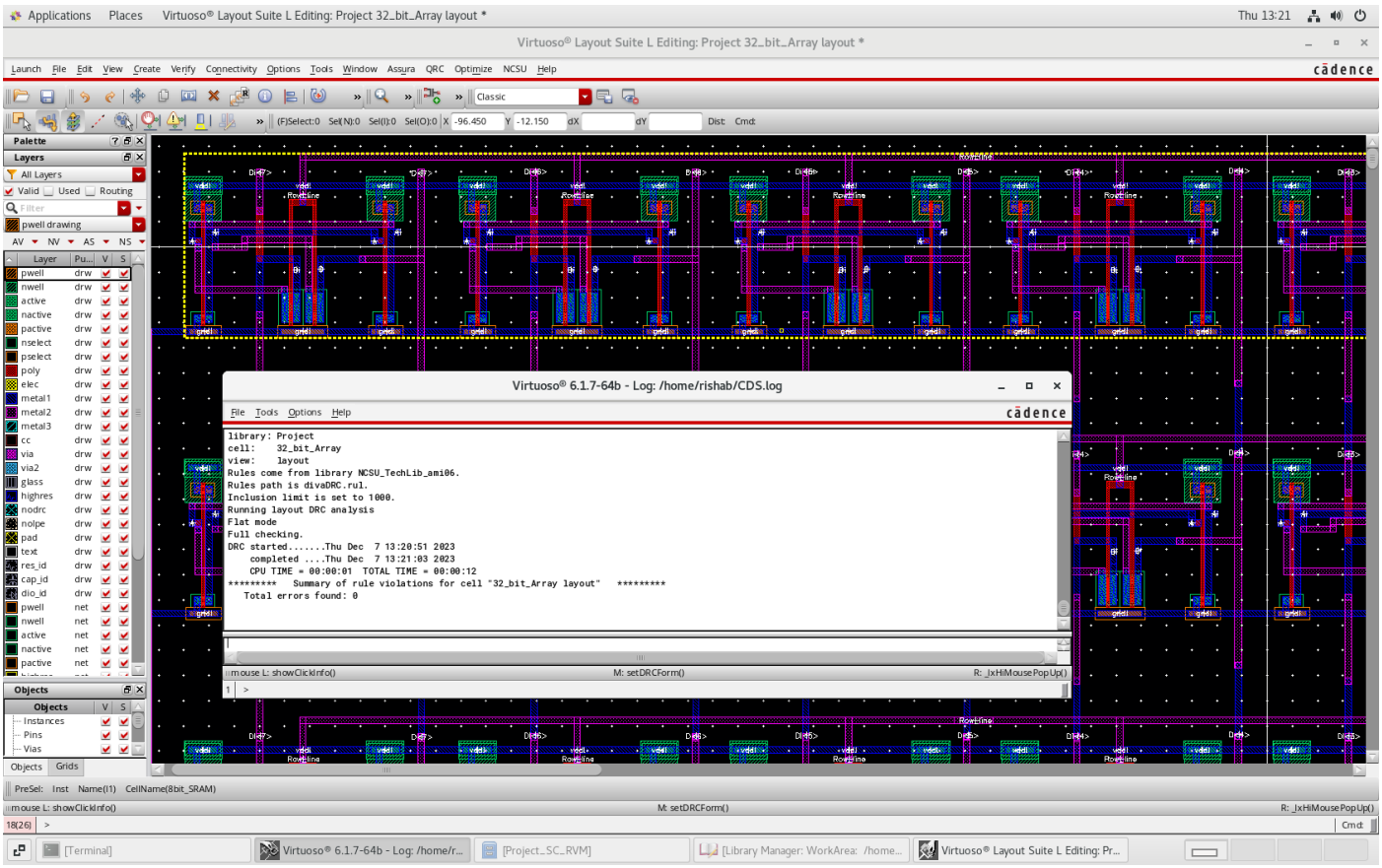


3.5.3 Layout

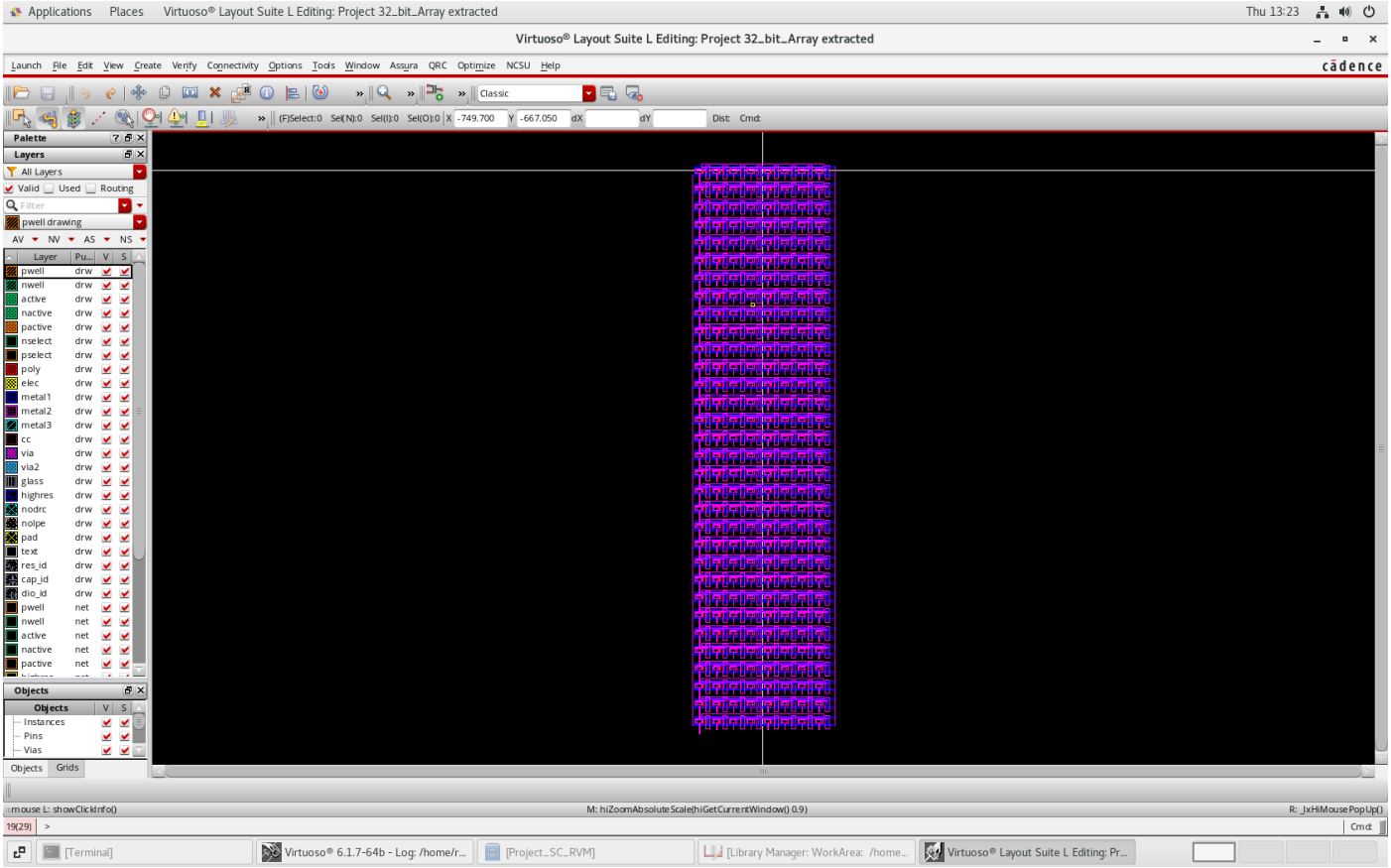

Applications Places Virtuoso® Layout Suite L Editing: Project 32_bit_Array layout
Thu 13:19
Virtuoso® Layout Suite L Editing: Project 32_bit_Array layout
Launch File Edit View Create Verify Connectivity Options Tools Window Assura QRC Optimize NCSU Help
cadence
Palette
Layers
Objects
[Terminal]
[Virtuoso® 6.1.7-64b - Log: /home/...
[Project_SC_RVM]
[Library Manager: WorkArea: /home...
Virtuoso® Layout Suite L Editing: Pr...

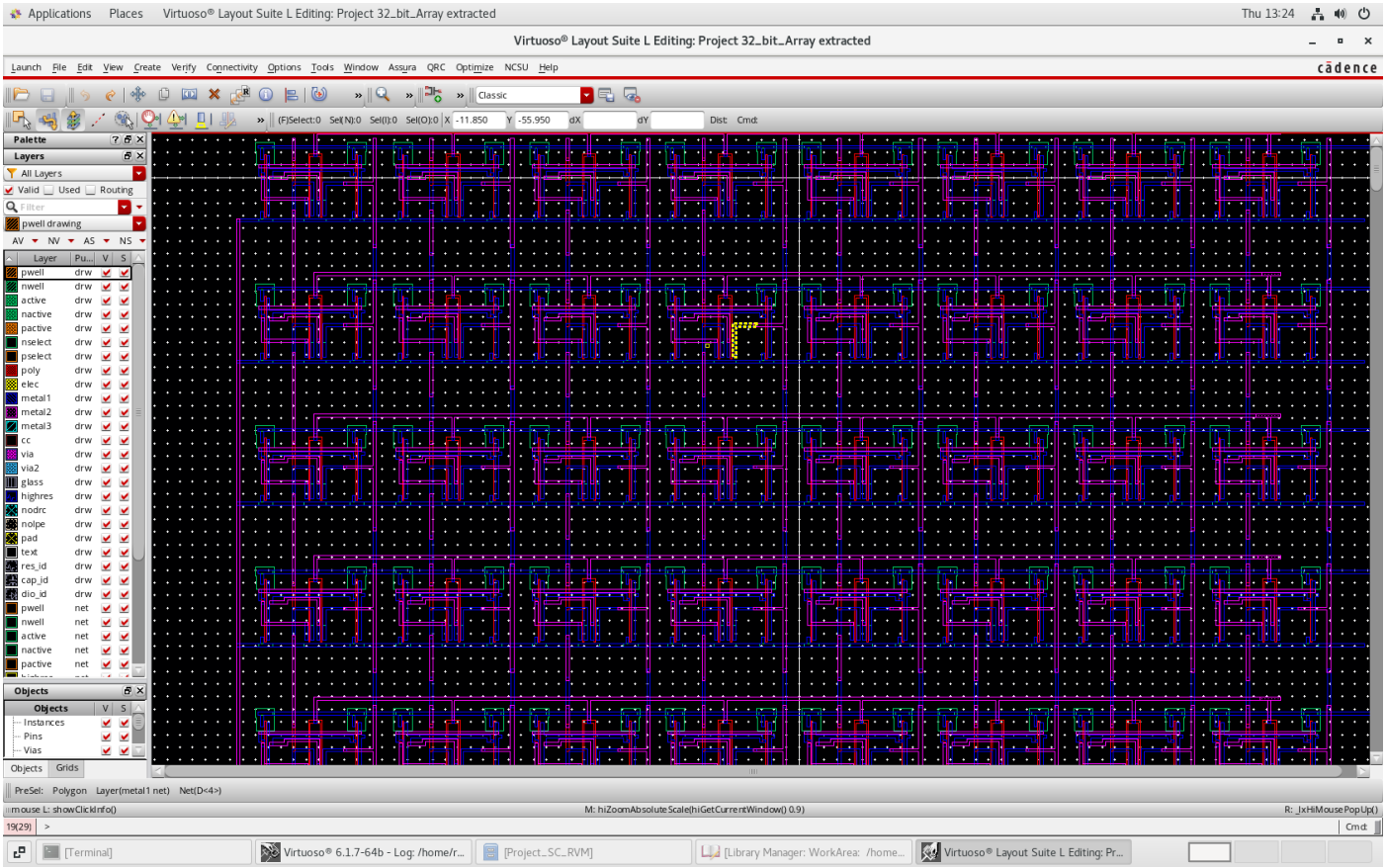

Applications Places Virtuoso® Layout Suite L Editing: Project 32_bit_Array layout
Thu 13:20
Virtuoso® Layout Suite L Editing: Project 32_bit_Array layout
Launch File Edit View Create Verify Connectivity Options Tools Window Assura QRC Optimize NCSU Help
cadence
Palette
Layers
Objects
[Terminal]
[Virtuoso® 6.1.7-64b - Log: /home/...
[Project_SC_RVM]
[Library Manager: WorkArea: /home...
Virtuoso® Layout Suite L Editing: Pr...

3.5.4 DRC

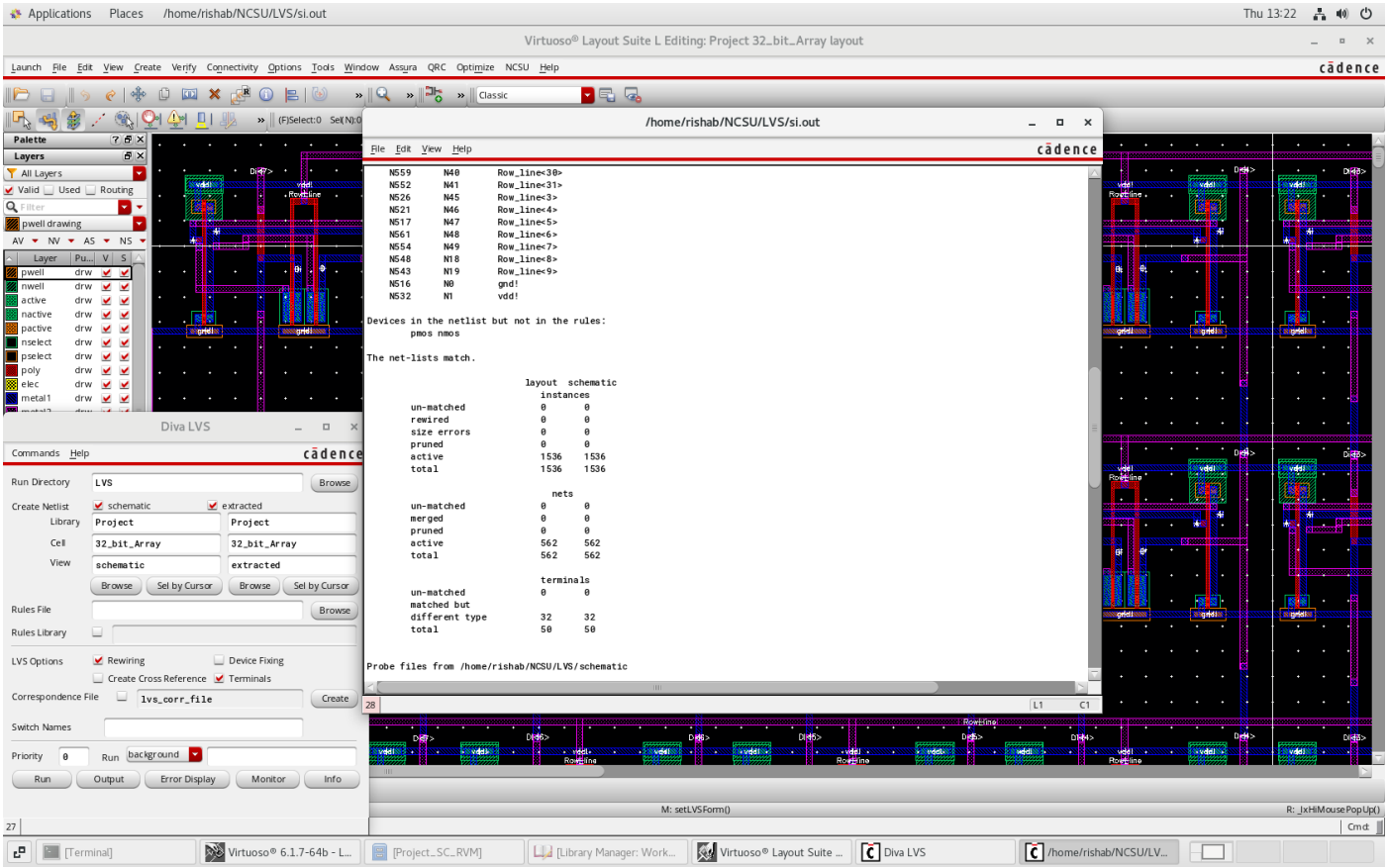


3.5.5 Extracted

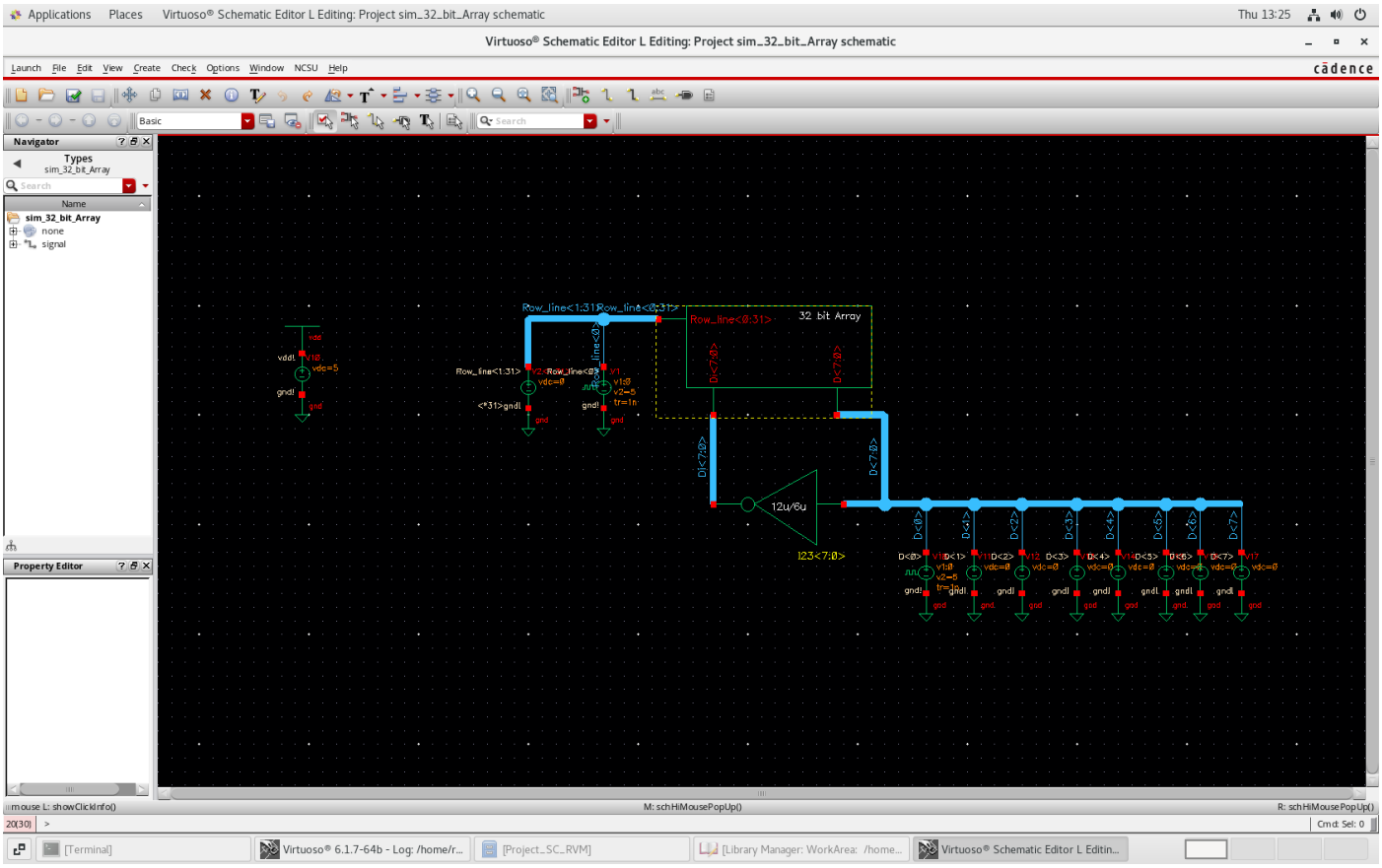


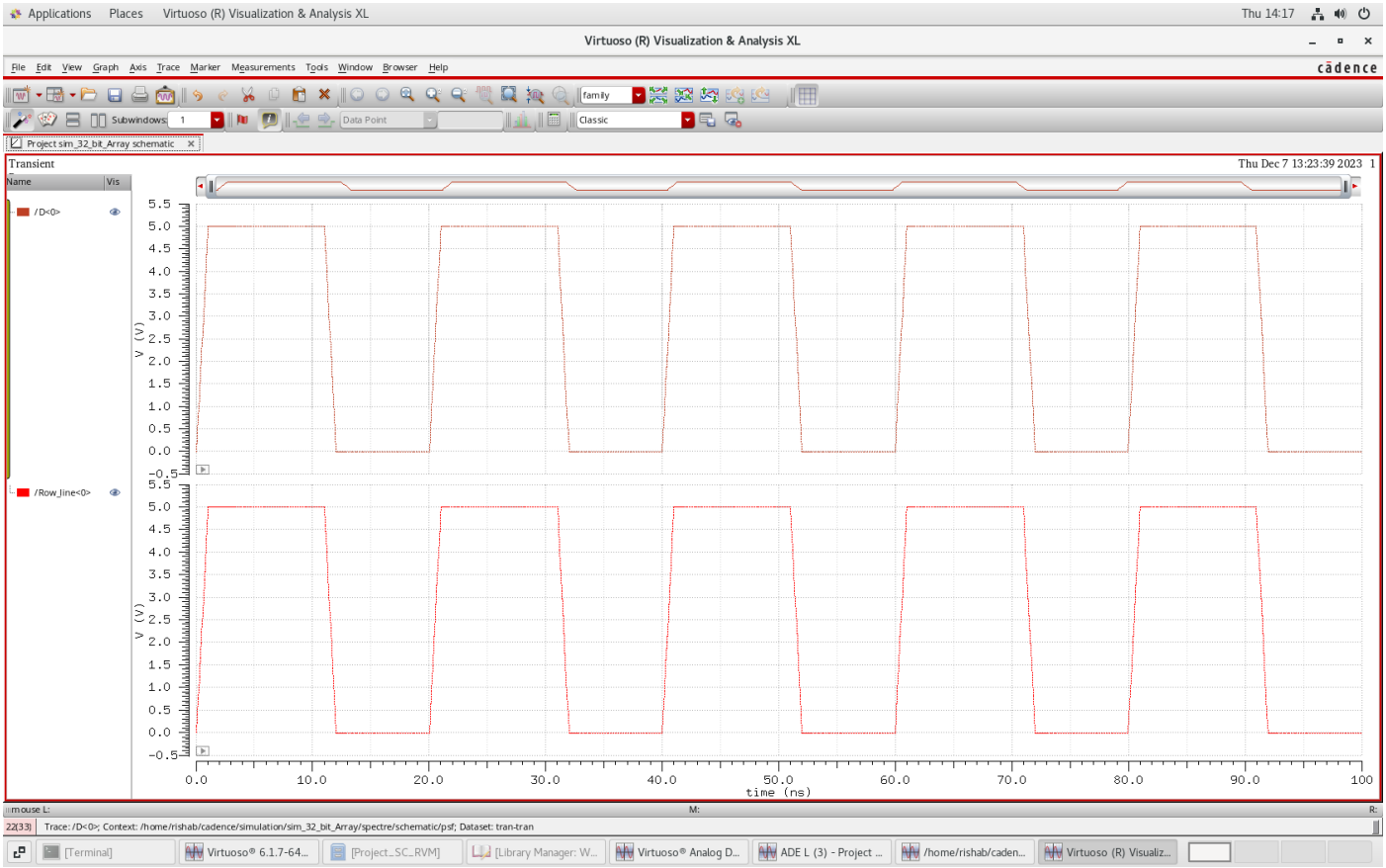

Applications Places Virtuoso® Layout Suite L Editing: Project 32_bit_Array extracted Thu 13:24
Virtuoso® Layout Suite L Editing: Project 32_bit_Array extracted
Launch File Edit View Create Verify Connectivity Options Tools Window Assura QRC Optimize NCSU Help
cadence
Palette
Layers
Objects
Instances
Pins
Vias
Objects Grids
Virtuoso® 6.1.7-64b - Log: /home/r...
[Project_SC_RVM]
[Library Manager: WorkArea: /home...
Virtuoso® Layout Suite L Editing: Pr...

3.5.6 LVS


Applications Places /home/rishab/NCSU/LVS/si.out Thu 13:22
Virtuoso® Layout Suite L Editing: Project 32_bit_Array layout
/home/rishab/NCSU/LVS/si.out
File Edit View Help
cadence
N559 N40 Row_line<30>
N552 N41 Row_line<31>
N526 N45 Row_line<3>
N521 N46 Row_line<4>
N517 N47 Row_line<5>
N561 N48 Row_line<6>
N554 N49 Row_line<7>
N548 N18 Row_line<8>
N543 N19 Row_line<9>
N516 N0 gnd!
N532 N1 vdd!
Devices in the netlist but not in the rules:
pmos nmos
The net-lists match.
layout schematic
instances
un-matched 0 0
rewired 0 0
size errors 0 0
pruned 0 0
active 1536 1536
total 1536 1536
nets
un-matched 0 0
merged 0 0
pruned 0 0
active 562 562
total 562 562
terminals
un-matched 0 0
matched but
different type 32 32
total 50 50
Probe files from /home/rishab/NCSU/LVS/schematic
Diva LVS
Commands Help
cadence
Run Directory LVS
Create Netlist schematic extracted
Library Project Project
Cell 32_bit_Array 32_bit_Array
View schematic extracted
Rules File
Rules Library
LVS Options Rewiring Device Fixing
Create Cross Reference Terminals
Correspondence File lvs_corr_file
Switch Names
Priority 0 Run background
Run Output Error Display Monitor Info
Virtuoso® 6.1.7-64b - L...
[Project_SC_RVM]
[Library Manager: Work...
Virtuoso® Layout Suite ...
Diva LVS
/home/rishab/NCSU/LV...

3.5.7 Simulation

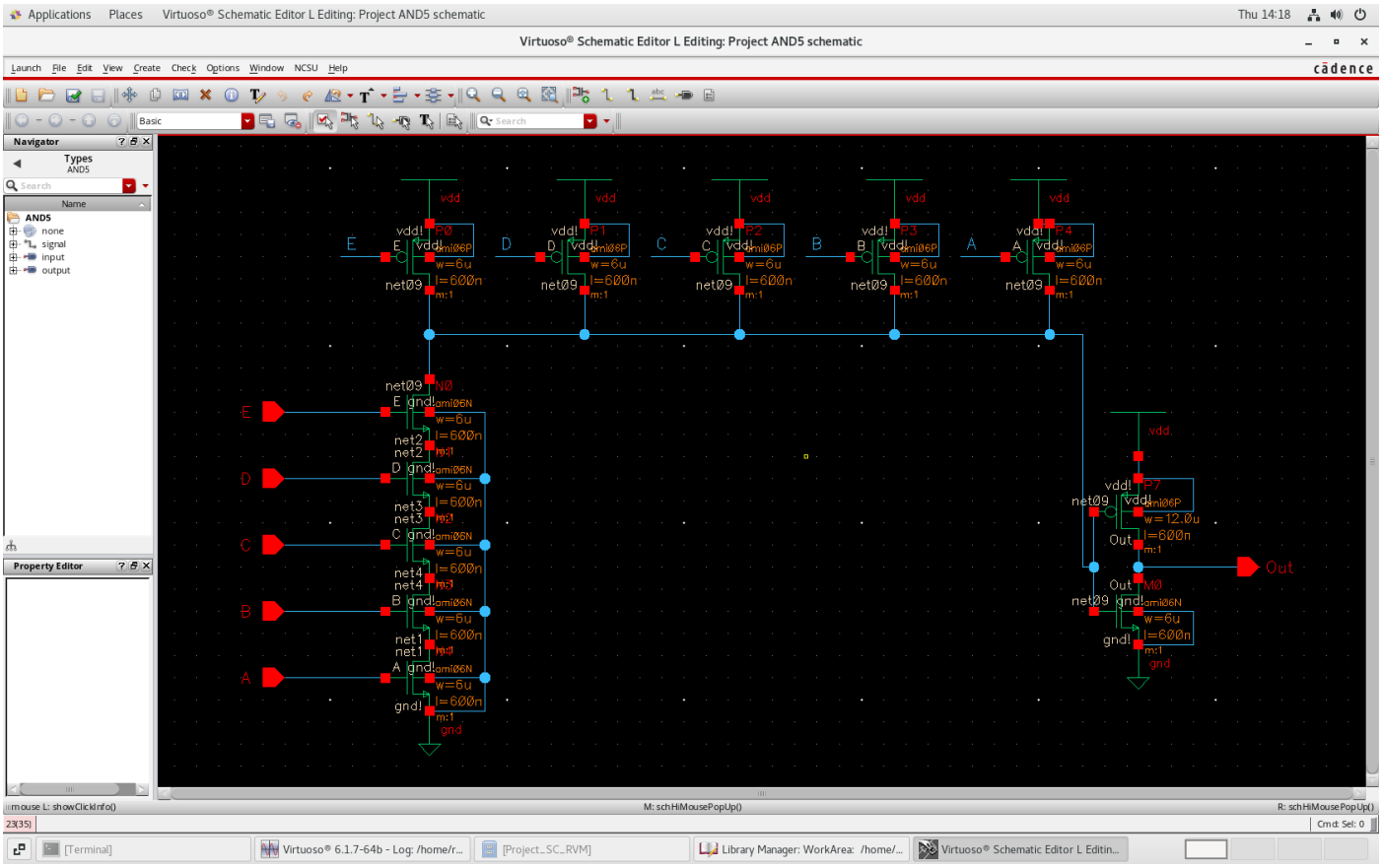




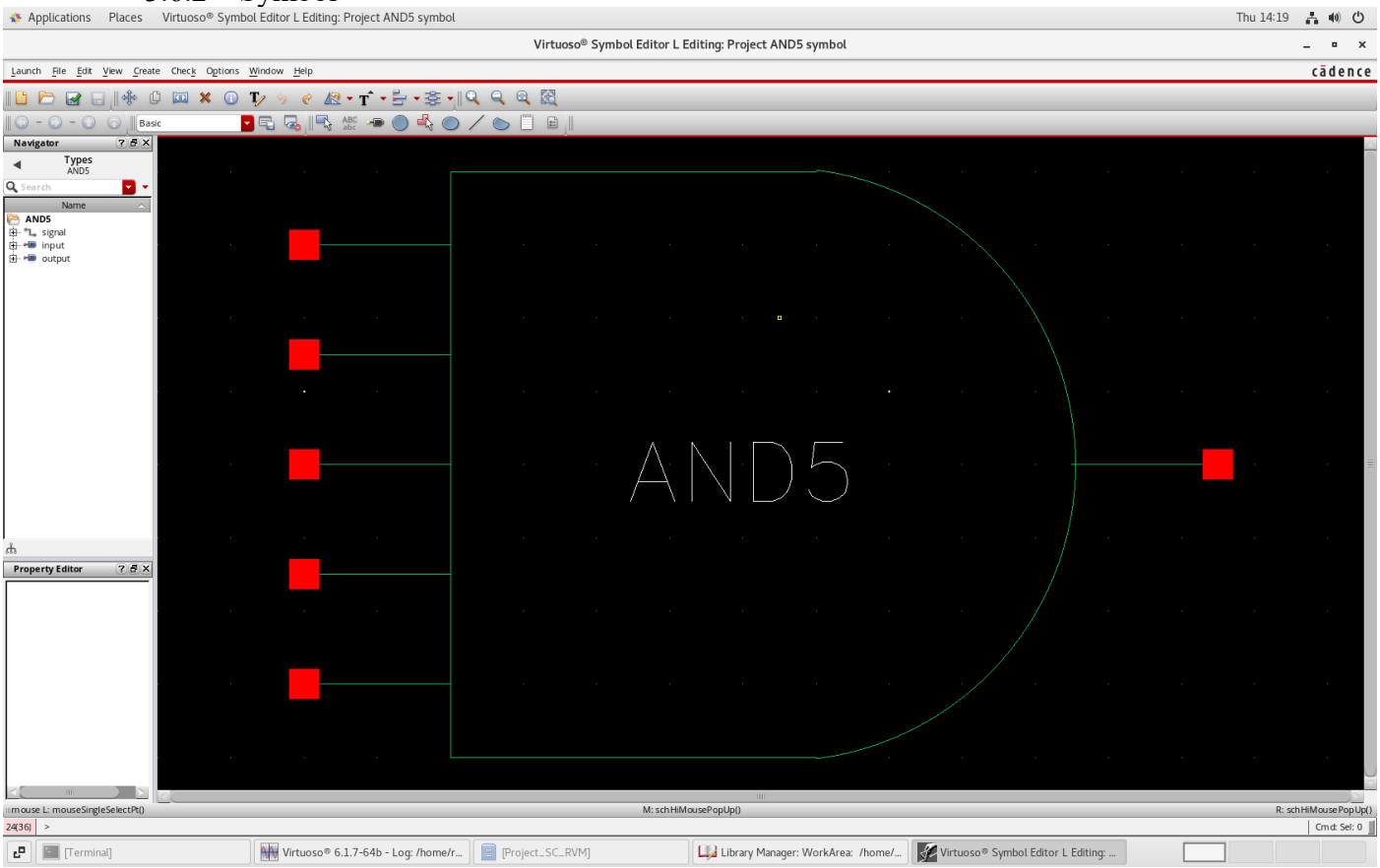
As you can see, the Data is written to D<0> of the first word when Row_line <0> is high.

3.6 5 input AND Gate

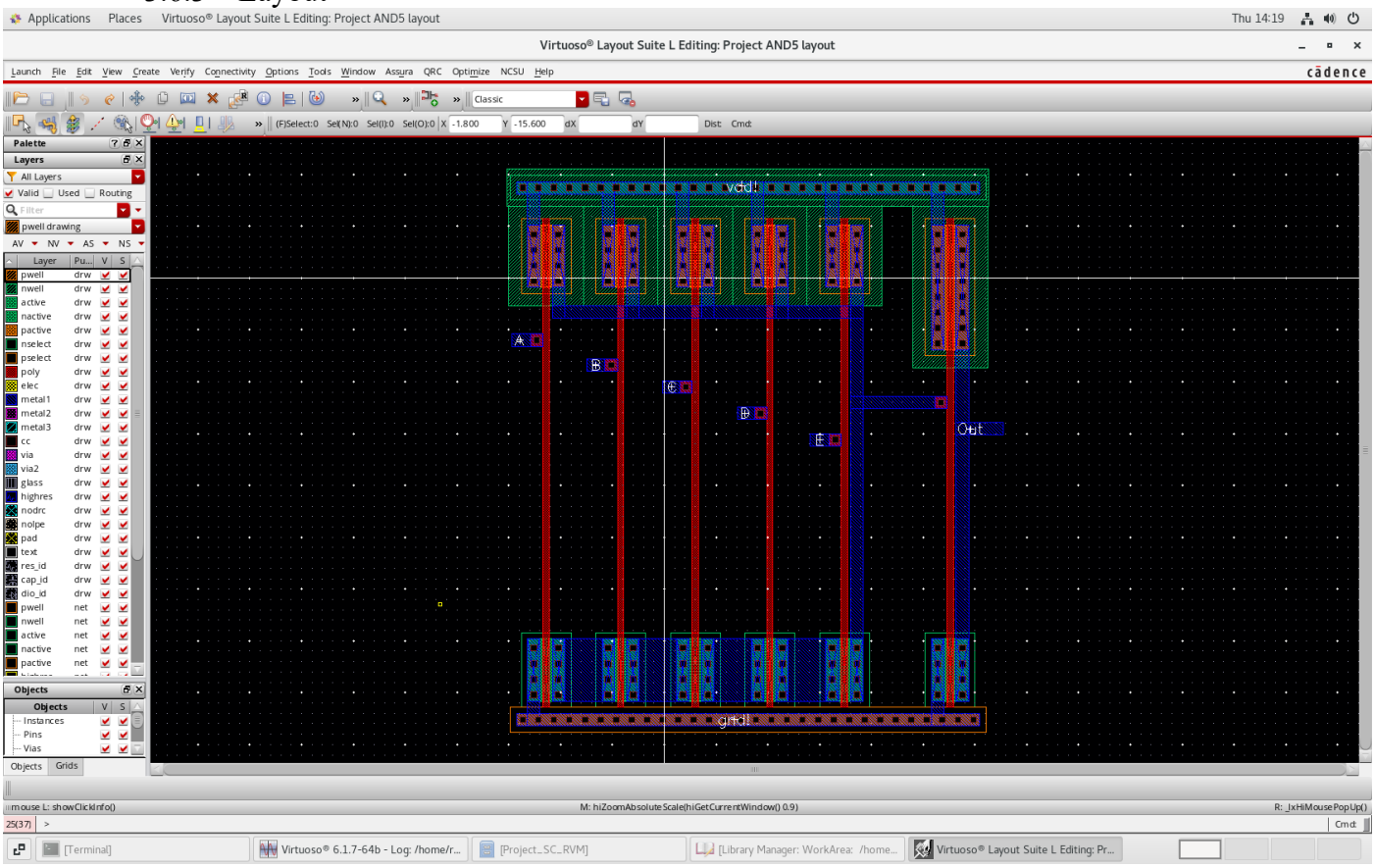
3.6.1 Schematic



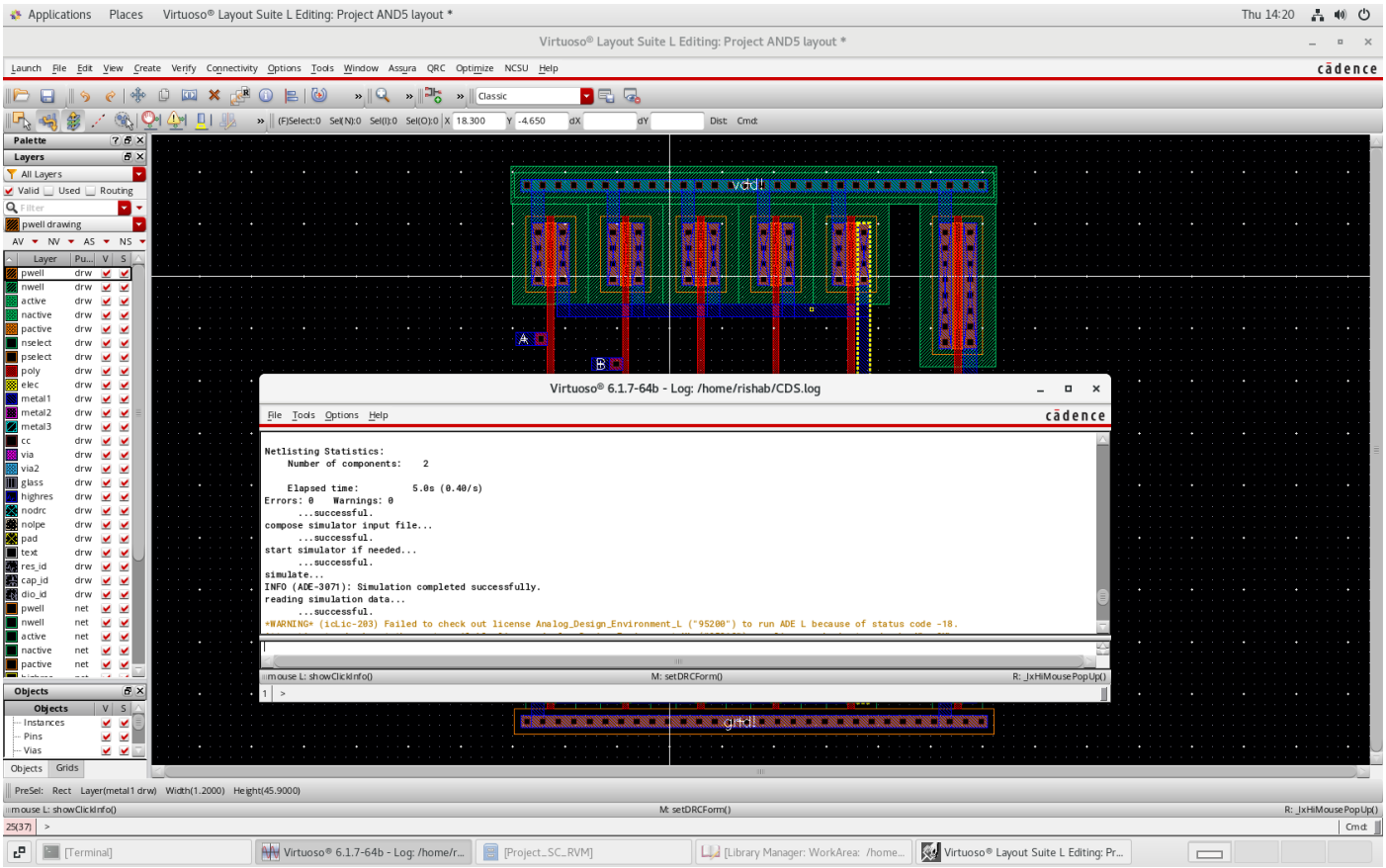
3.6.2 Symbol



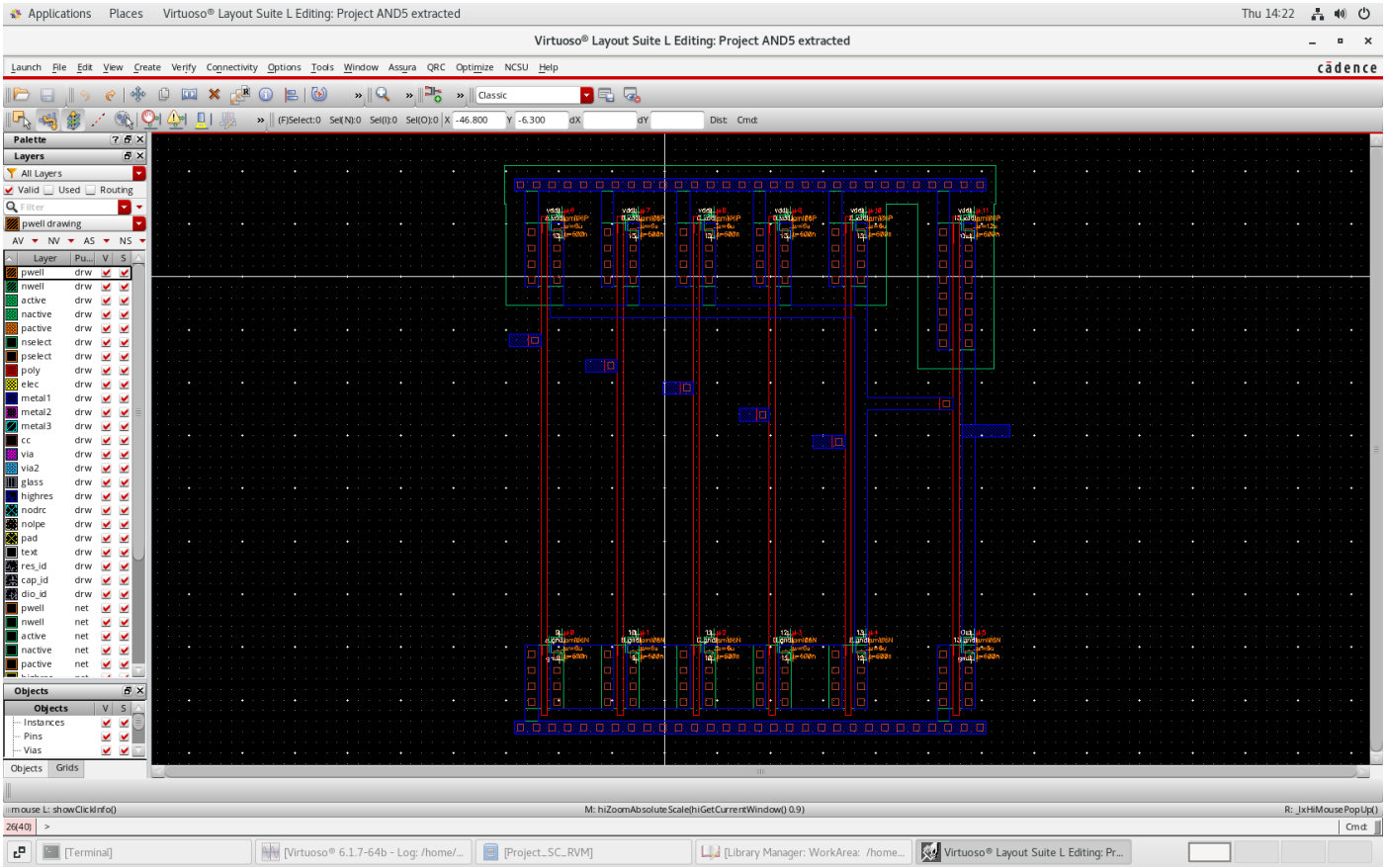
3.6.3 Layout



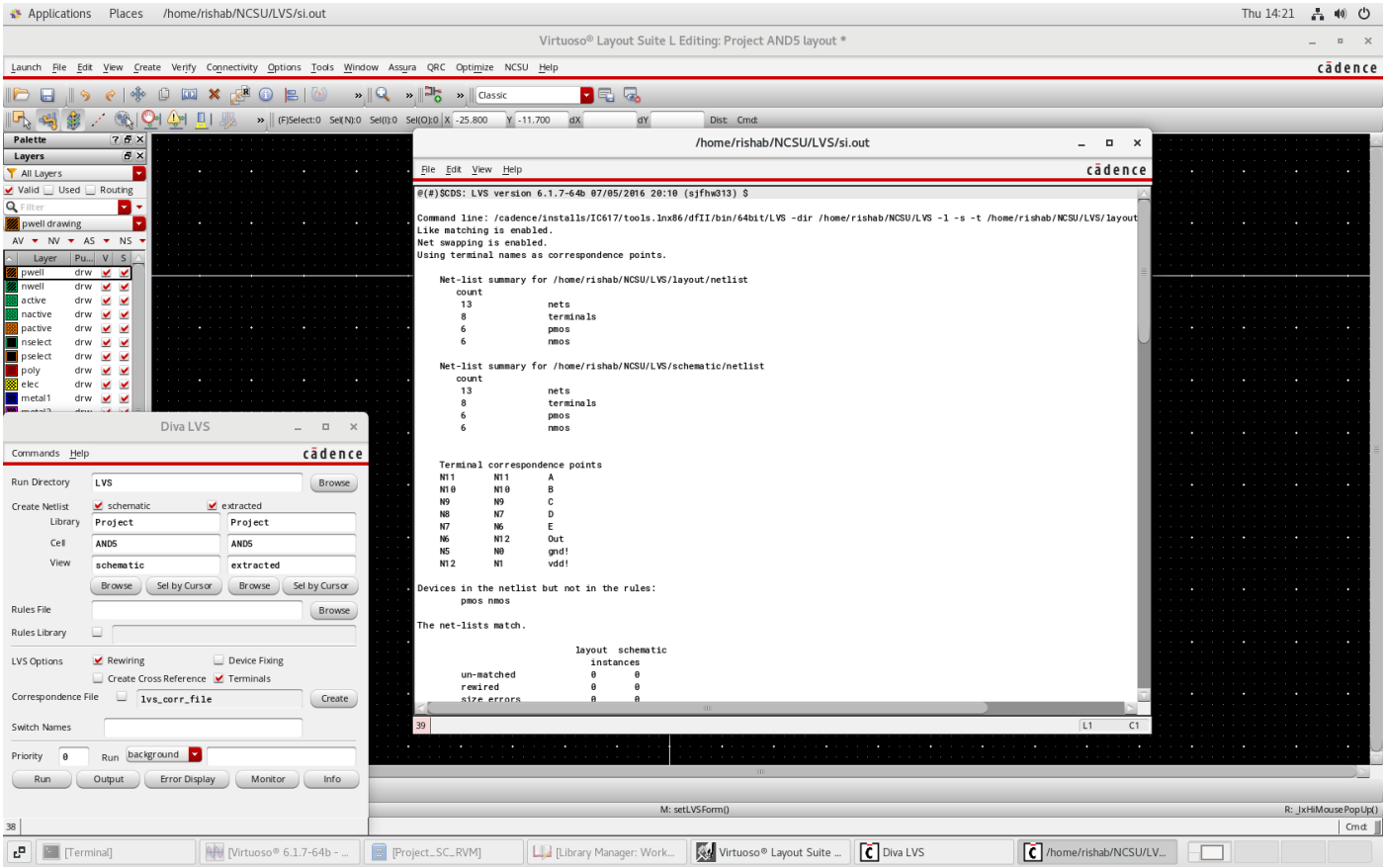
3.6.4 DRC



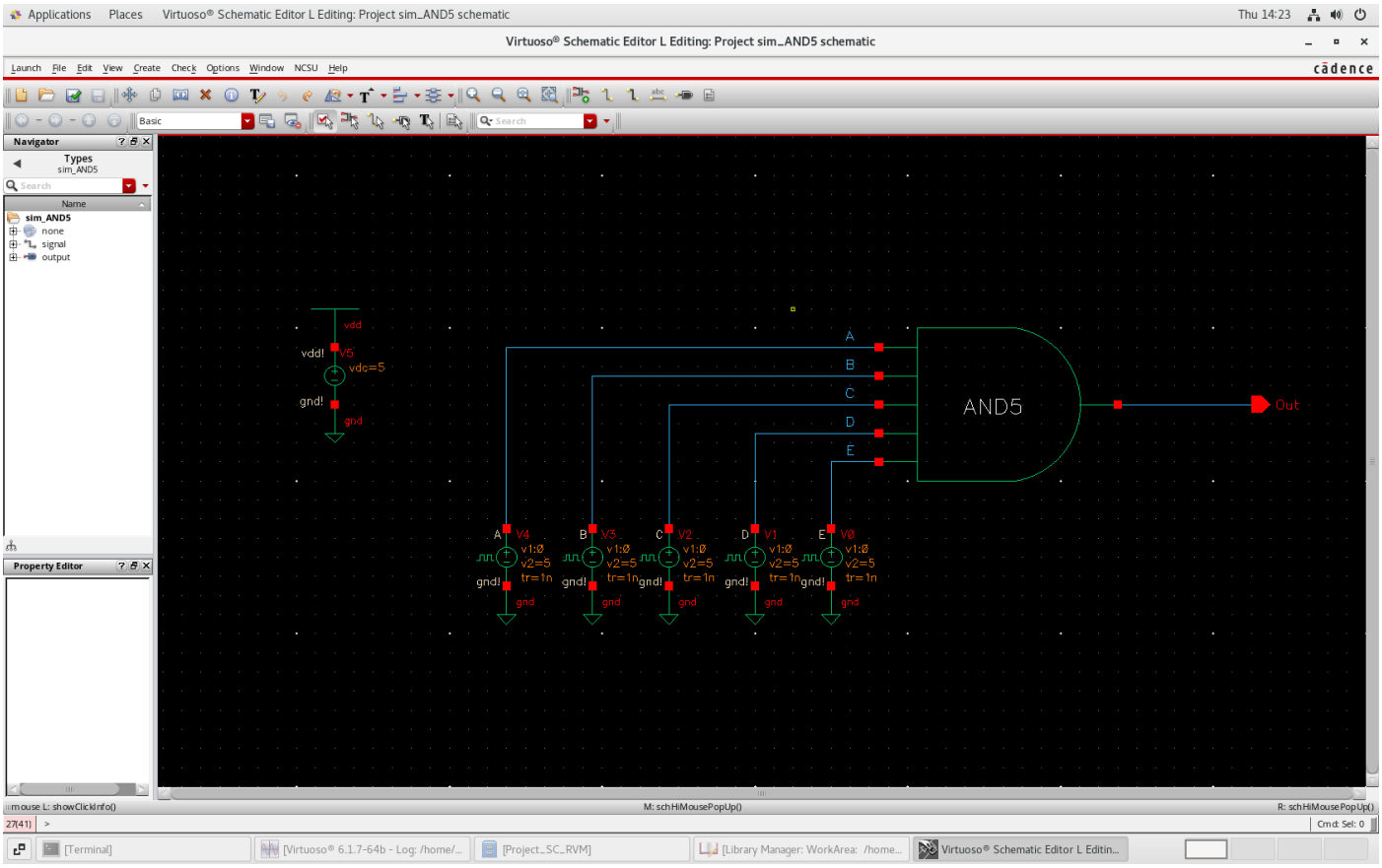
3.6.5 Extracted

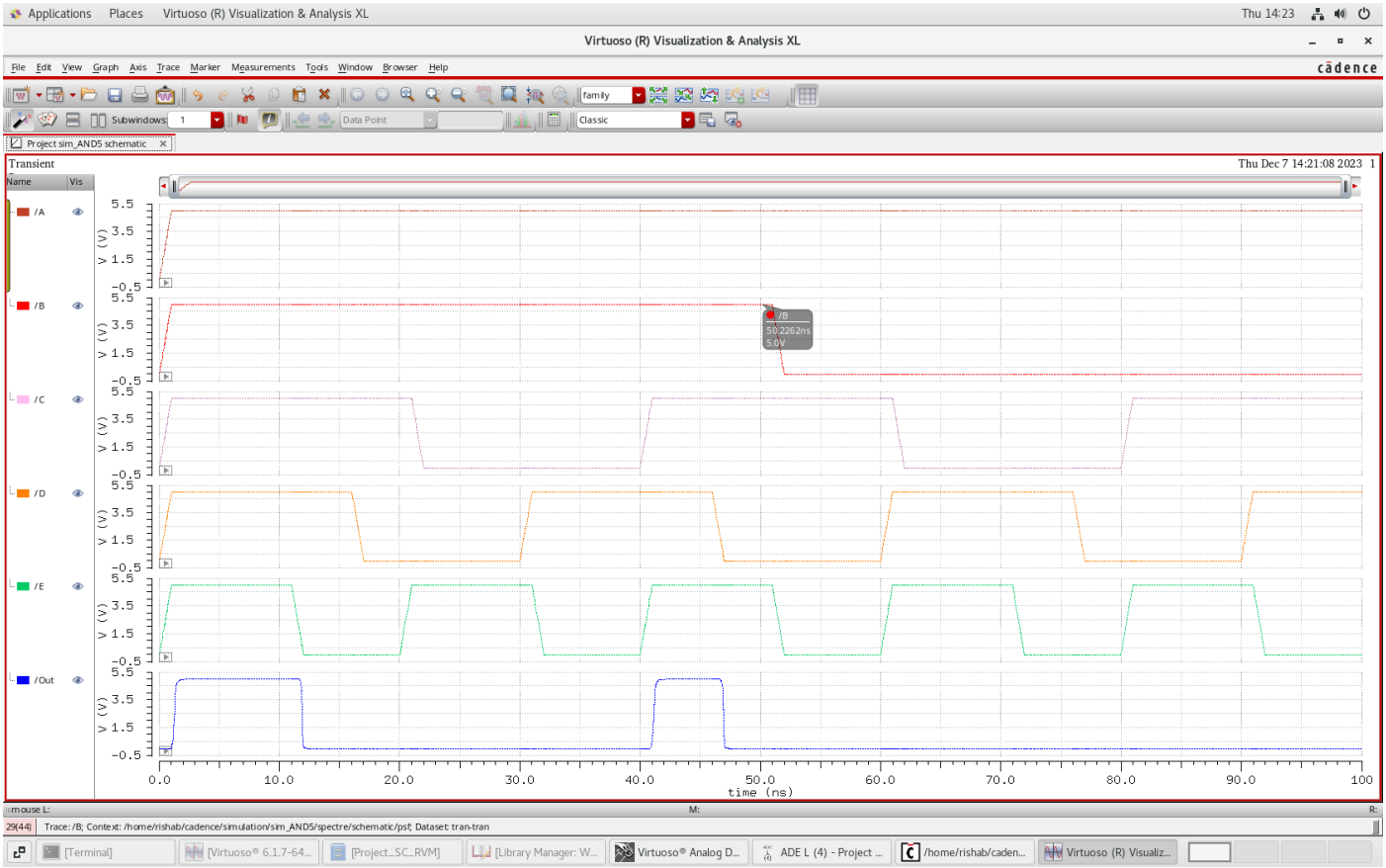


3.6.6 LVS



3.6.7 Simulation

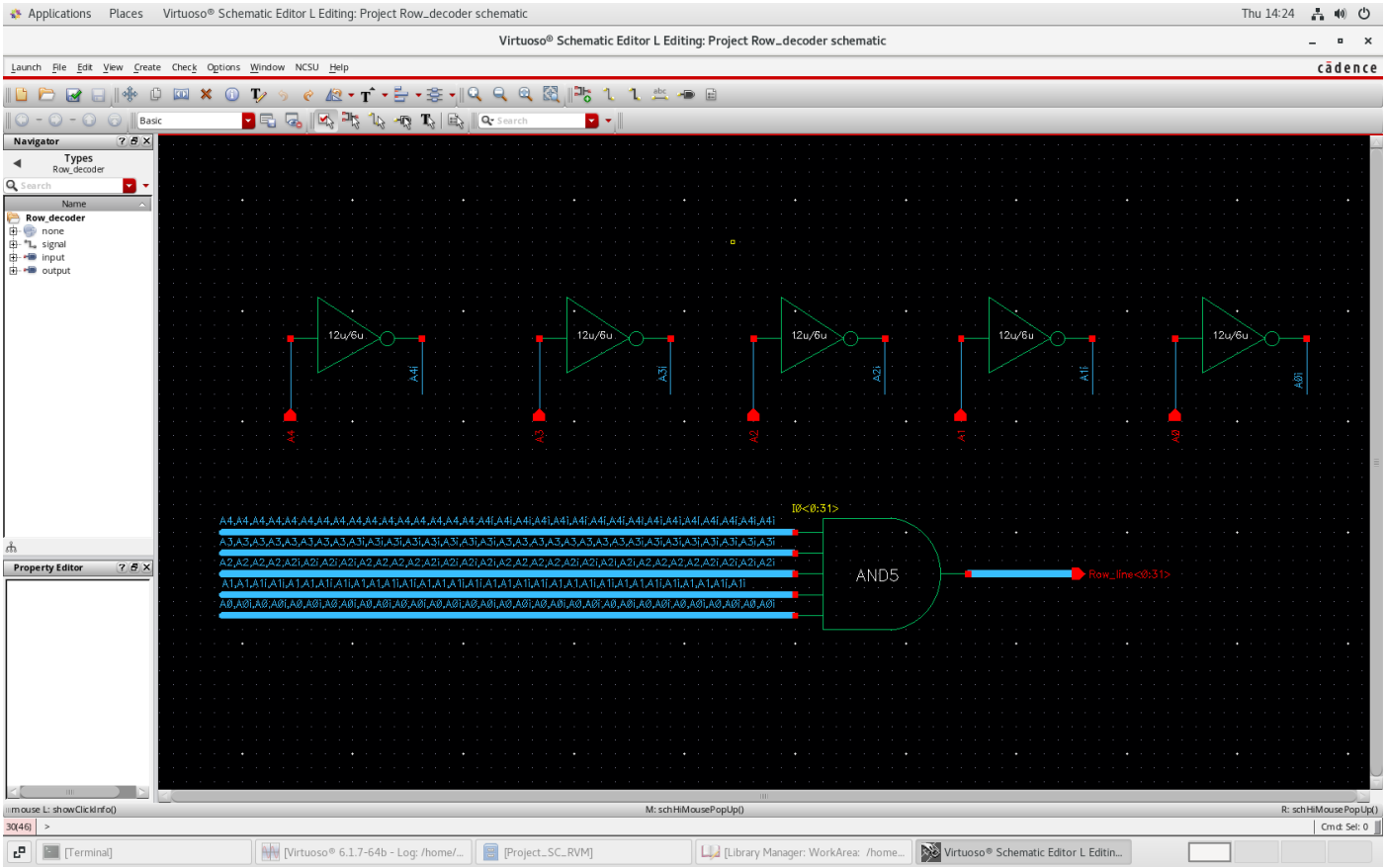




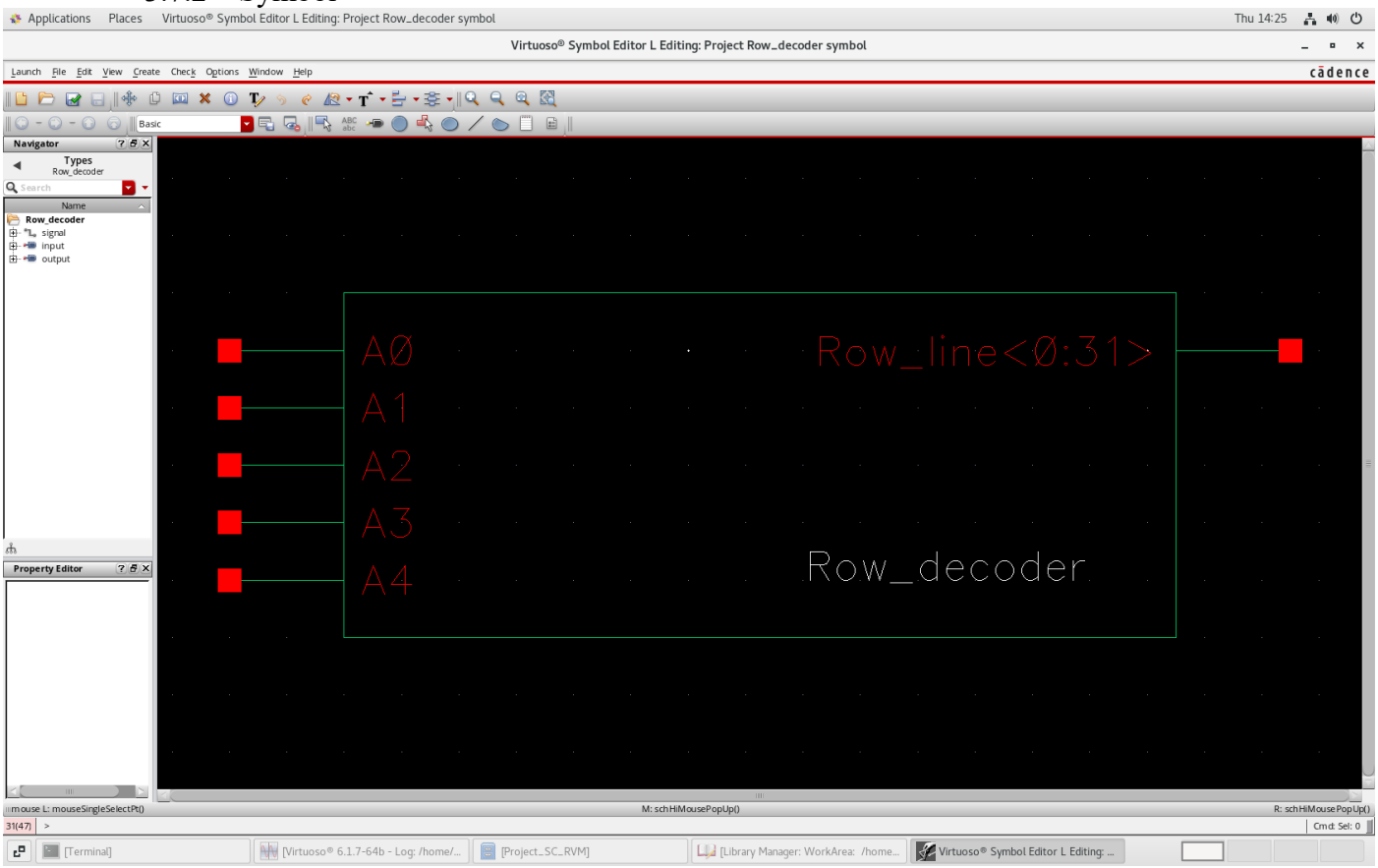
As observed, the AND gate is operating as intended, producing an output of 1 only when all inputs are set to 1.

3.7 Row Decoder

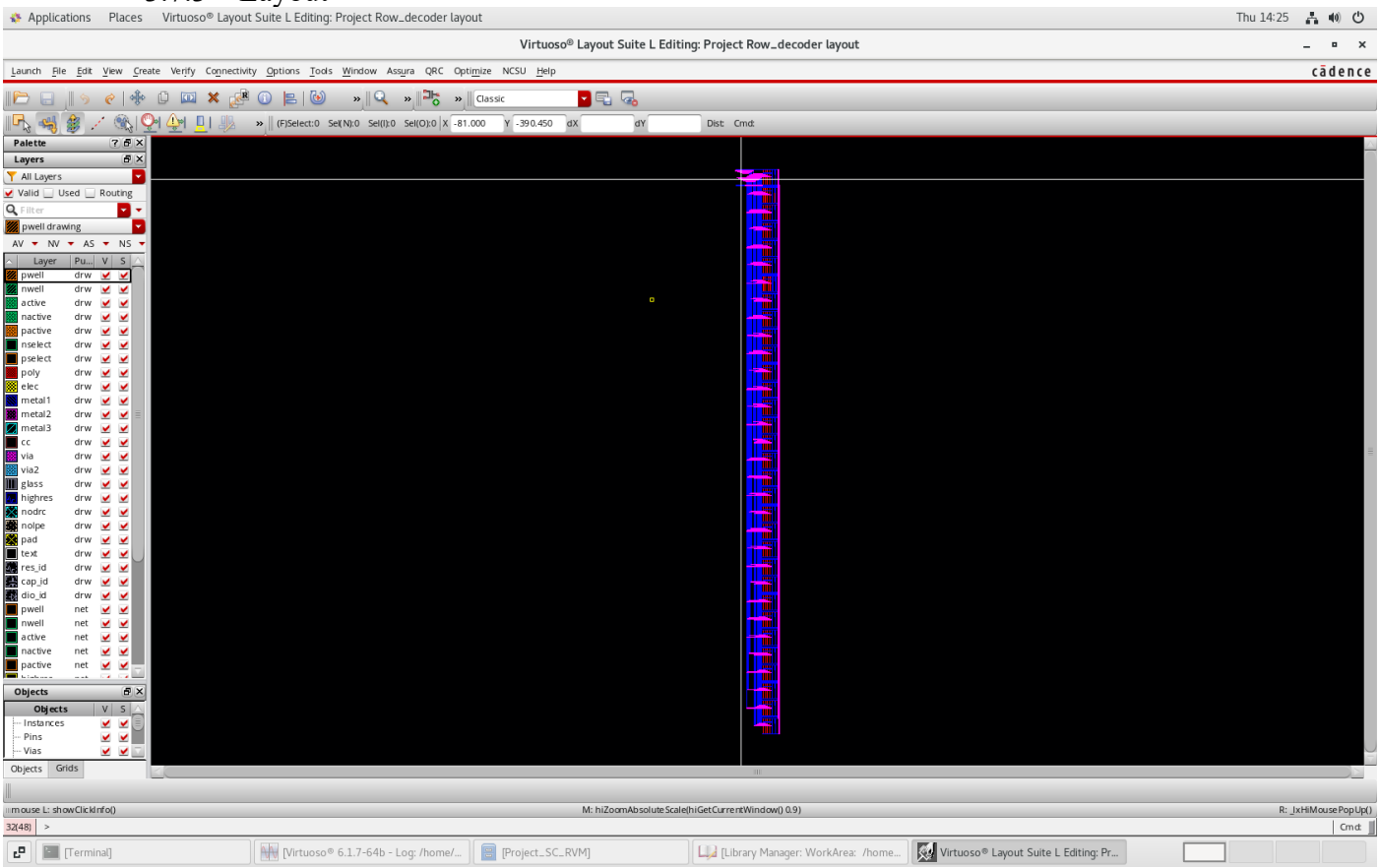
3.7.1 Schematic

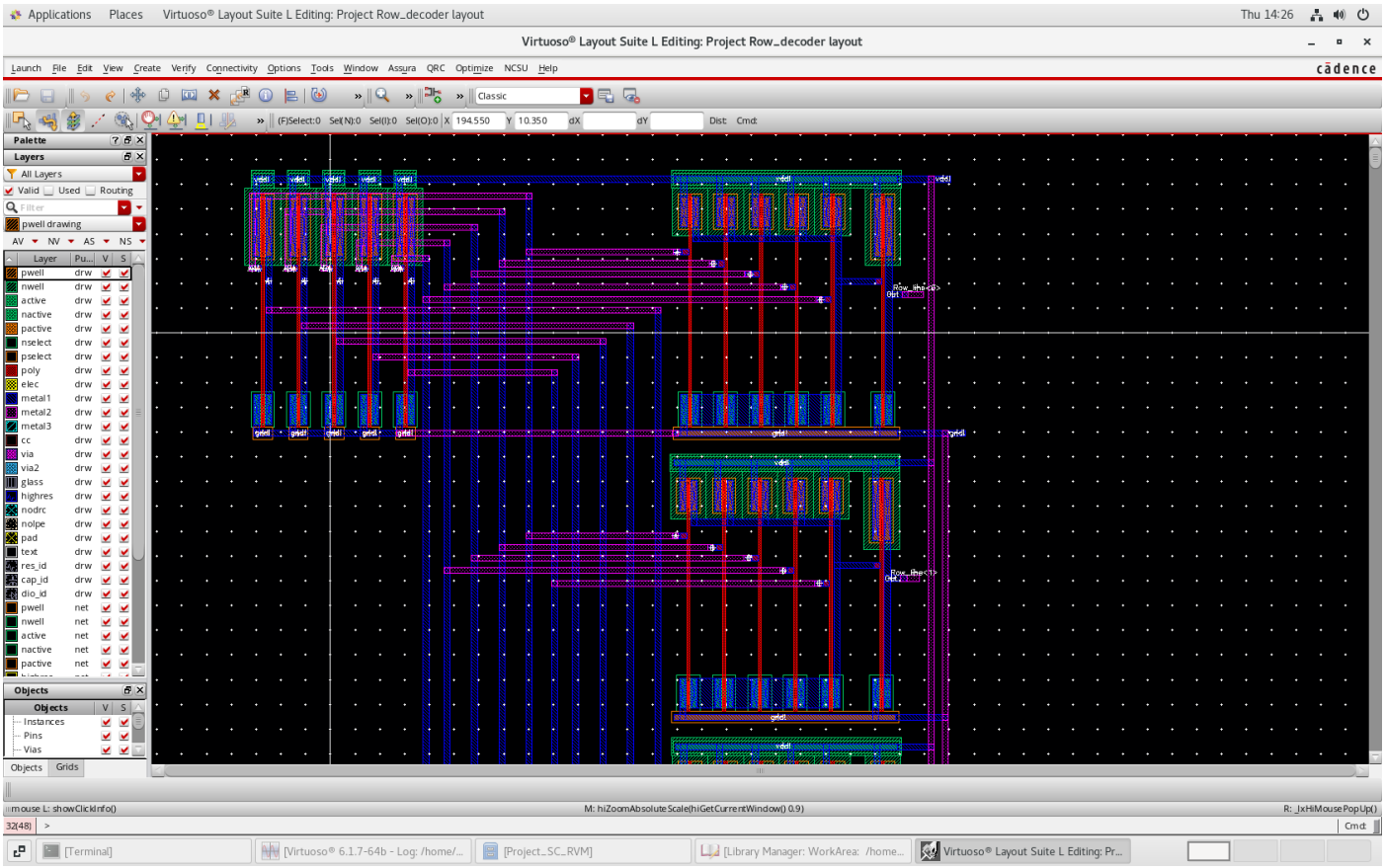


3.7.2 Symbol

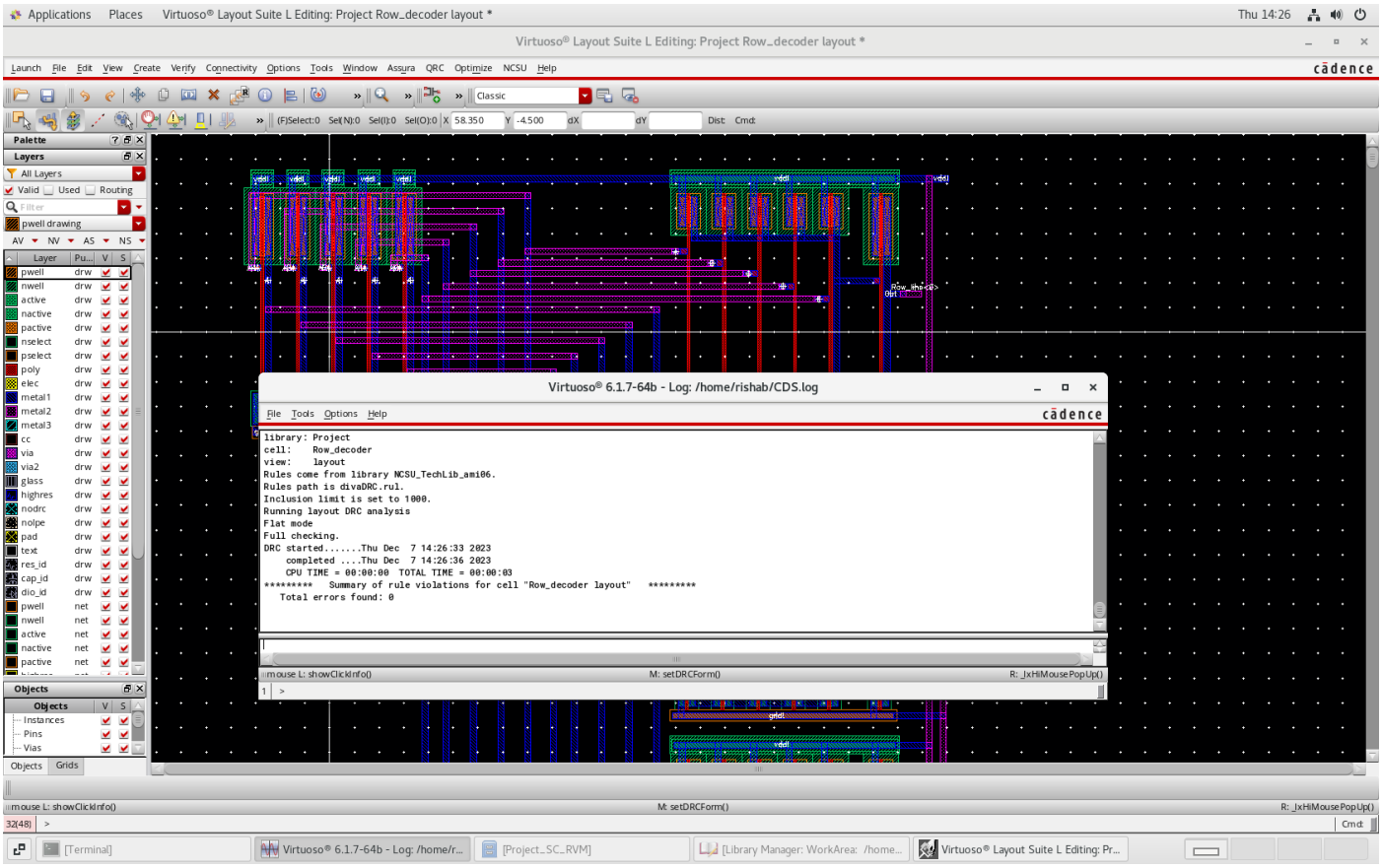


3.7.3 Layout

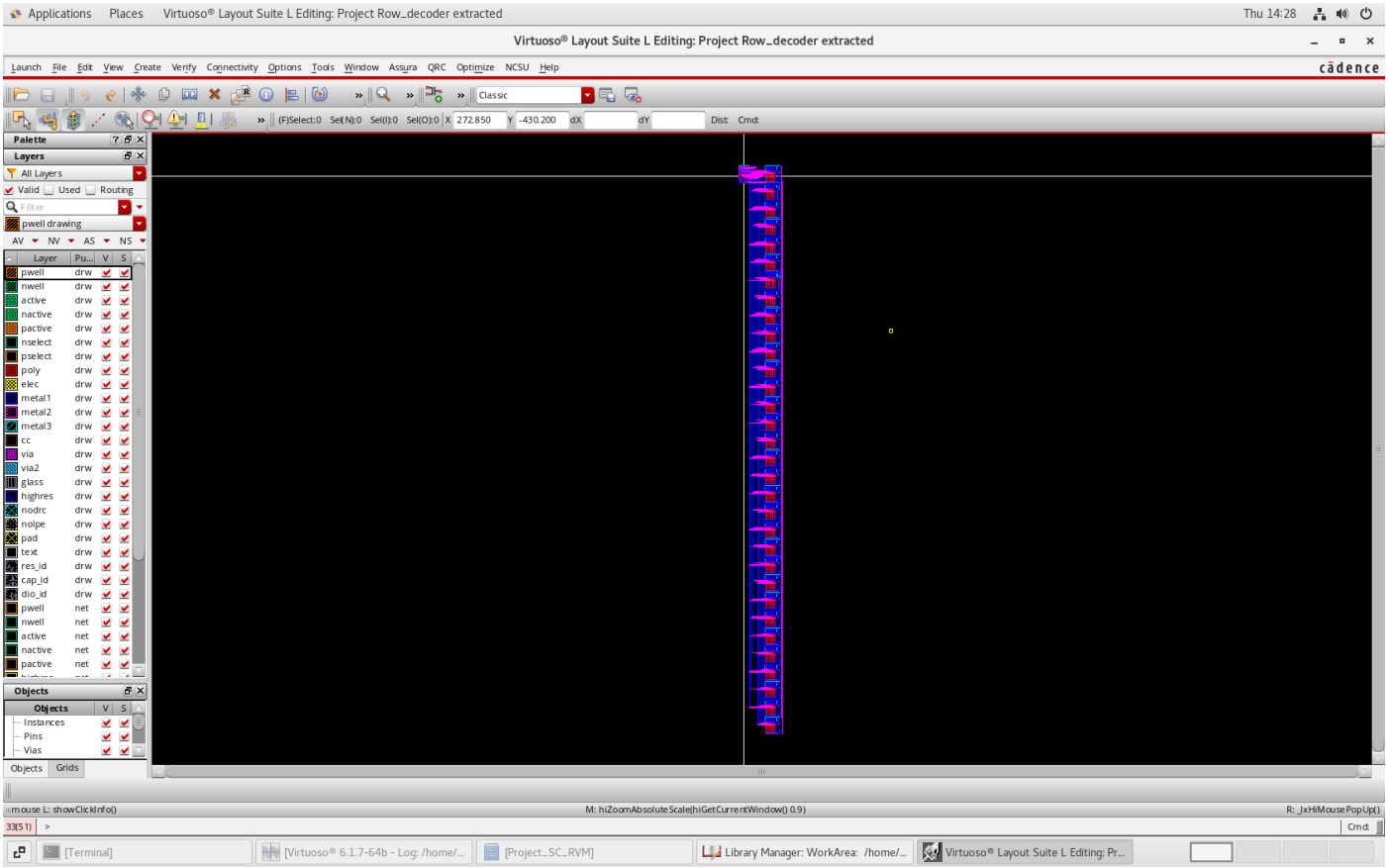

Applications Places Virtuoso® Layout Suite L Editing: Project Row_decoder layout
Thu 14:25
Virtuoso® Layout Suite L Editing: Project Row_decoder layout
Launch File Edit View Create Verify Connectivity Options Tools Window Assura QRC Optimize NCSU Help
cadence
Palette
Layers
All Layers
Valid Used Routing
pwell drawing
Objects
Instances
Pins
Vias
Objects Grids
mouse L: showClickInfo()
M: hiZoomAbsoluteScale(hiGetCurrentWindow() 0.9)
R: _lxHiMousePopUp()
[Terminal]
[Virtuoso® 6.1.7-64b - Log: /home/...
[Project_SC_RVM]
[Library Manager: WorkArea: /home...
Virtuoso® Layout Suite L Editing: Pr...

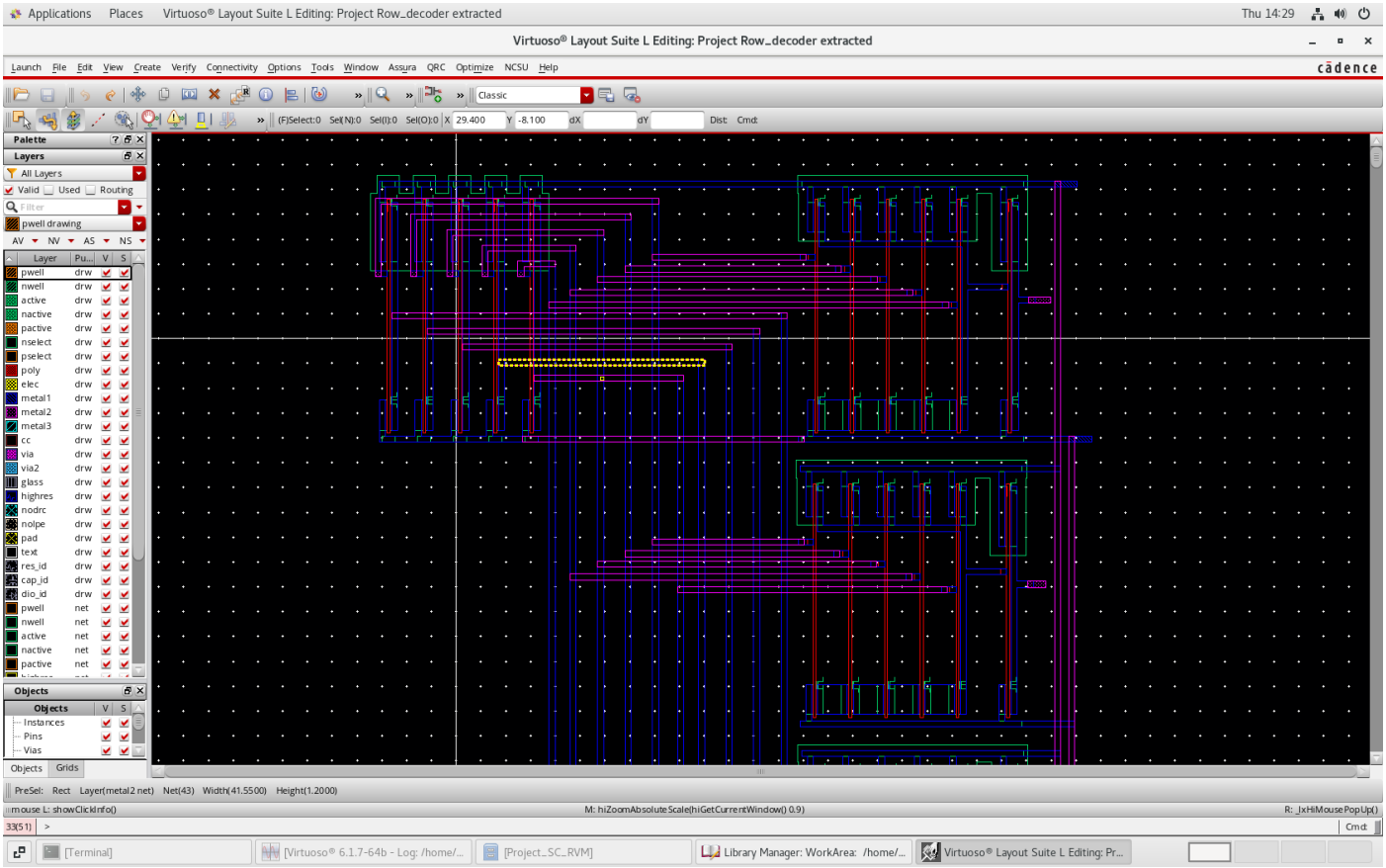

Applications Places Virtuoso® Layout Suite L Editing: Project Row_decoder layout
Thu 14:26
Virtuoso® Layout Suite L Editing: Project Row_decoder layout
Launch File Edit View Create Verify Connectivity Options Tools Window Assura QRC Optimize NCSU Help
cadence
Palette
Layers
All Layers
Valid Used Routing
pwell drawing
Objects
Instances
Pins
Vias
Objects Grids
mouse L: showClickInfo()
M: hiZoomAbsoluteScale(hiGetCurrentWindow() 0.9)
R: _lxHiMousePopUp()
[Terminal]
[Virtuoso® 6.1.7-64b - Log: /home/...
[Project_SC_RVM]
[Library Manager: WorkArea: /home...
Virtuoso® Layout Suite L Editing: Pr...

3.7.4 DRC

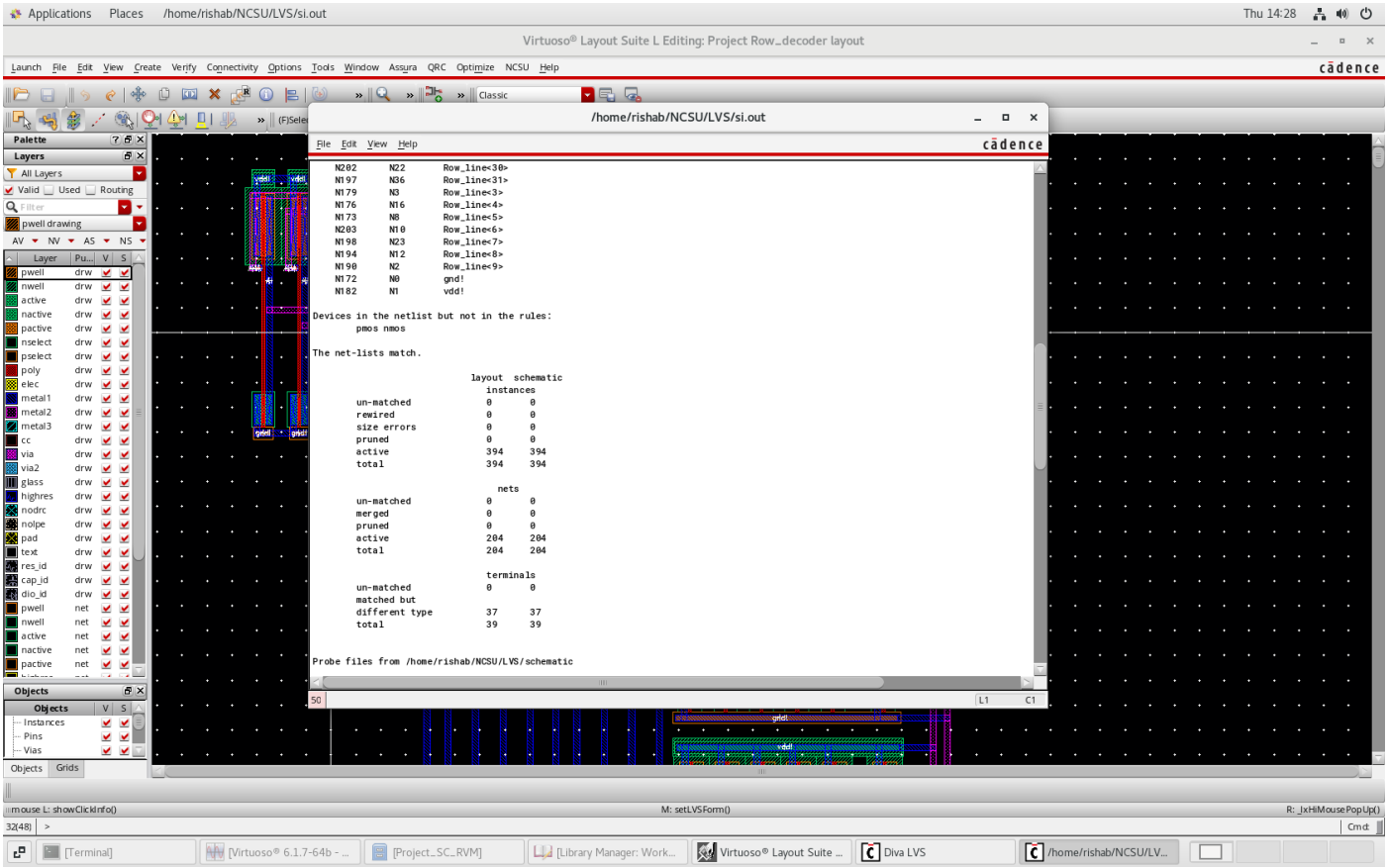


3.7.5 Extracted

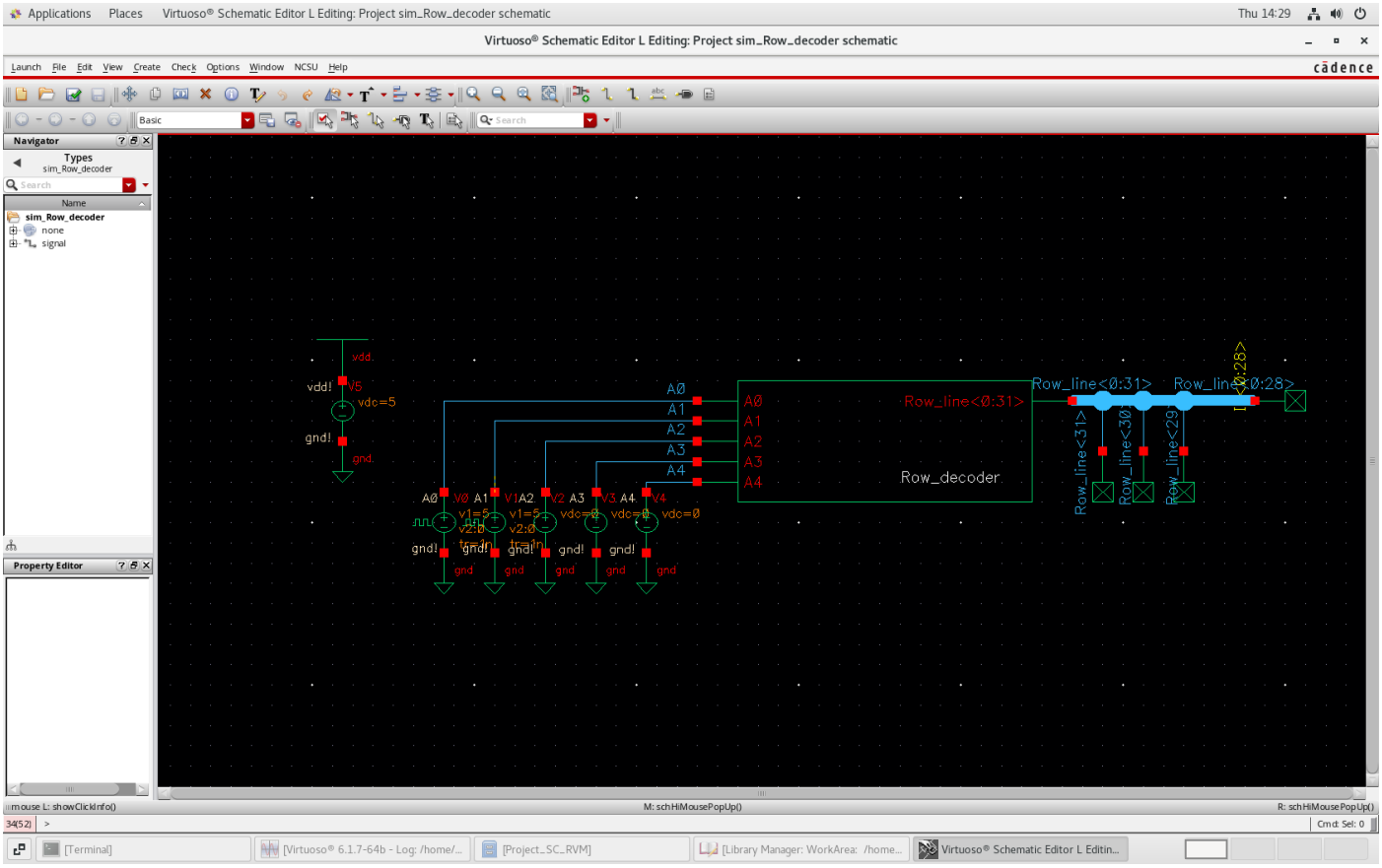



Applications Places Virtuoso® Layout Suite L Editing: Project Row_decoder extracted
Thu 14:29
Virtuoso® Layout Suite L Editing: Project Row_decoder extracted
Launch File Edit View Create Verify Connectivity Options Tools Window Assura QRC Optimize NCSU Help
cadence
Palette
Layers
Objects
[Terminal]
[Virtuoso® 6.1.7-64b - Log: /home/...
[Project_SC_RVM]
Library Manager: WorkArea: /home/...
Virtuoso® Layout Suite L Editing: Pr...

3.7.6 LVS


Applications Places /home/rishab/NCSU/LVS/si.out
Thu 14:28
Virtuoso® Layout Suite L Editing: Project Row_decoder layout
/home/rishab/NCSU/LVS/si.out
File Edit View Help
cadence
N202 N22 Row_line<30>
N197 N36 Row_line<31>
N179 N3 Row_line<3>
N176 N16 Row_line<4>
N173 N8 Row_line<5>
N203 N10 Row_line<6>
N198 N23 Row_line<7>
N194 N12 Row_line<8>
N190 N2 Row_line<9>
N172 N0 gnd!
N182 N1 vdd!
Devices in the netlist but not in the rules:
pmos nmos
The net-lists match.
layout schematic
instances
un-matched 0 0
rewired 0 0
size errors 0 0
pruned 0 0
active 394 394
total 394 394
nets
un-matched 0 0
merged 0 0
pruned 0 0
active 204 204
total 204 204
terminals
un-matched 0 0
matched but
different type 37 37
total 39 39
Probe files from /home/rishab/NCSU/LVS/schematic
Diva LVS

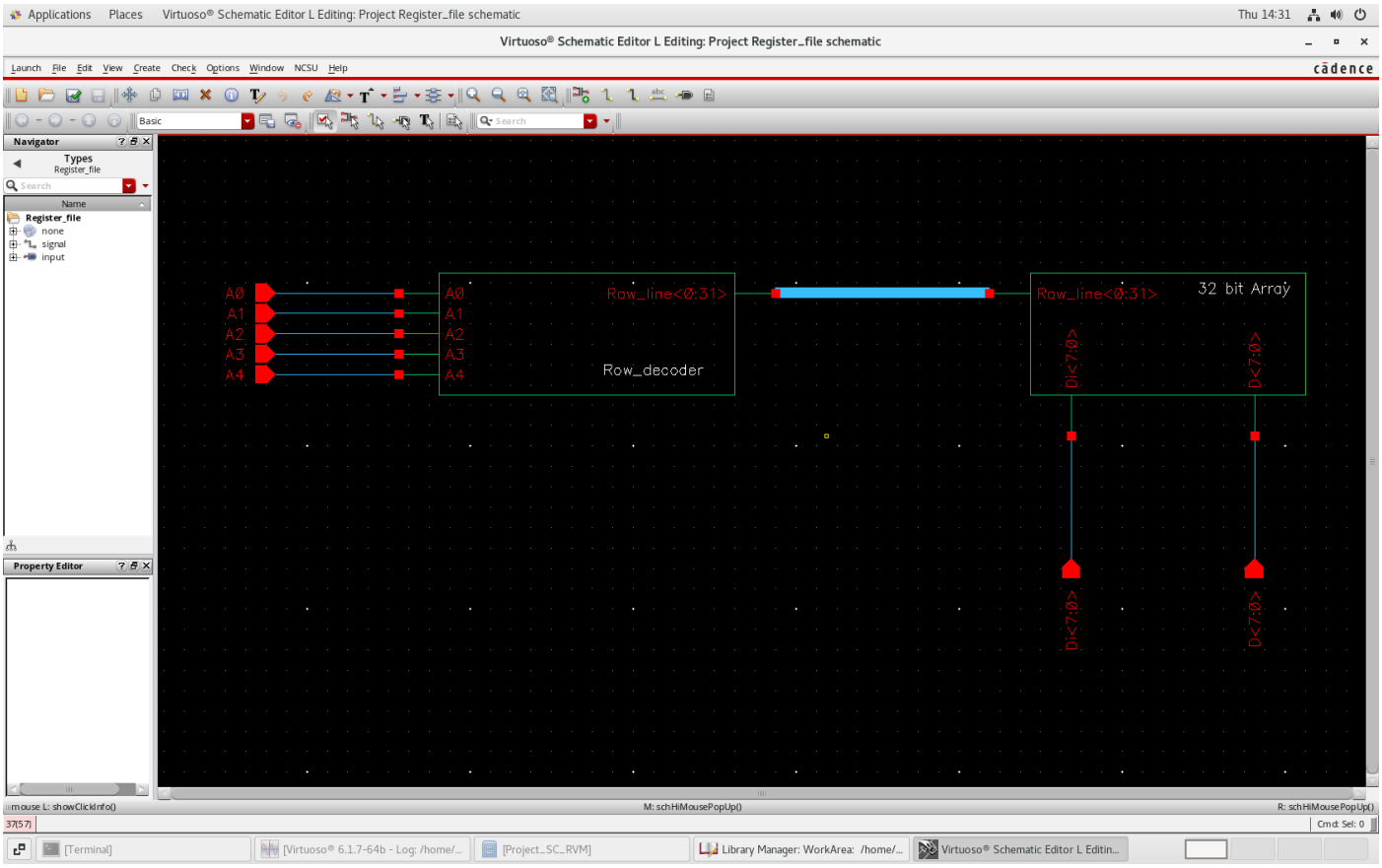
3.7.7 Simulation



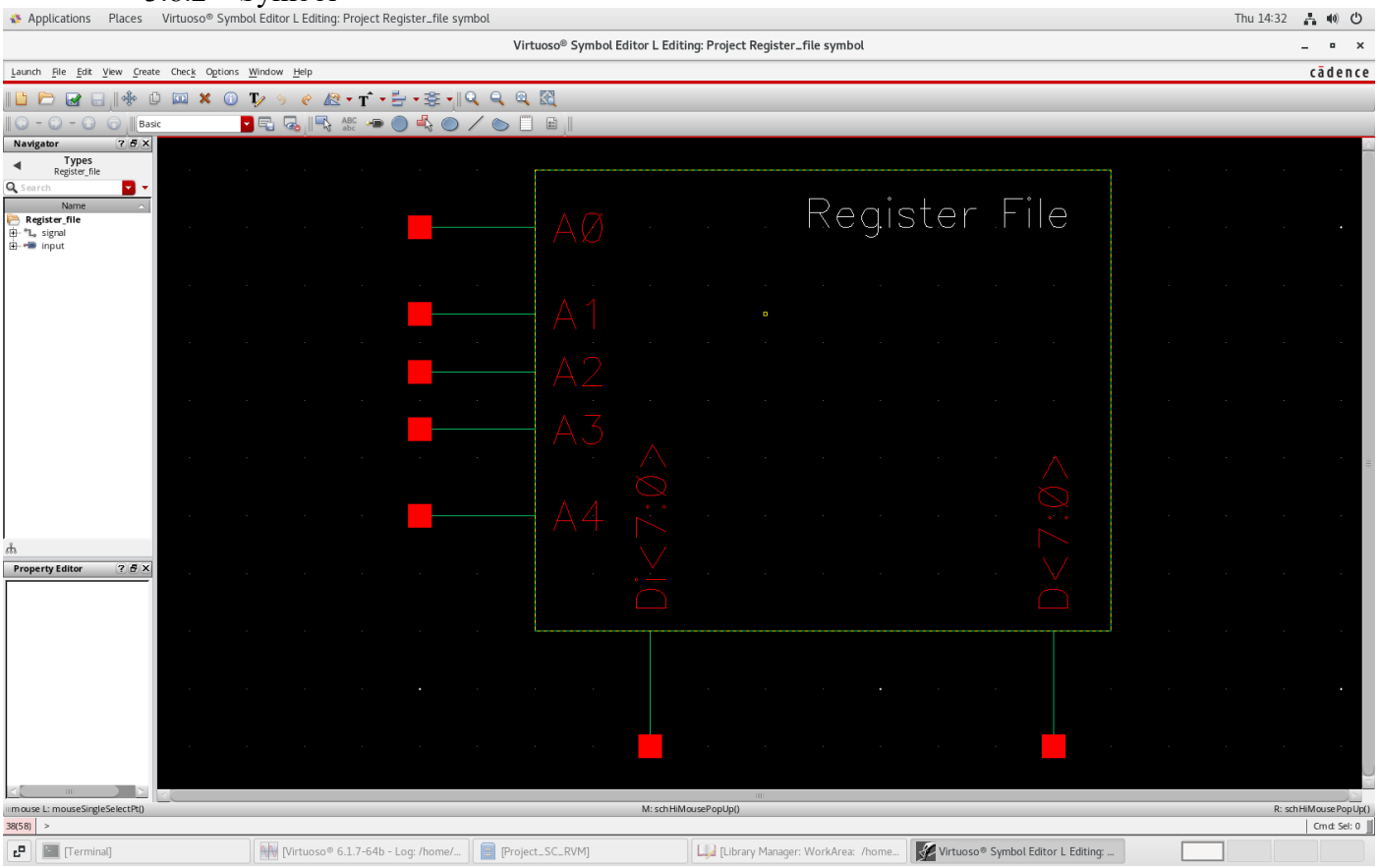
As evident, the row decoder is operating correctly. For instance, at address "00000," Row_line<31> is active, at address "00001," Row_line<30> is active, and at address "00010," Row_line<29> is active.

3.8 Register File

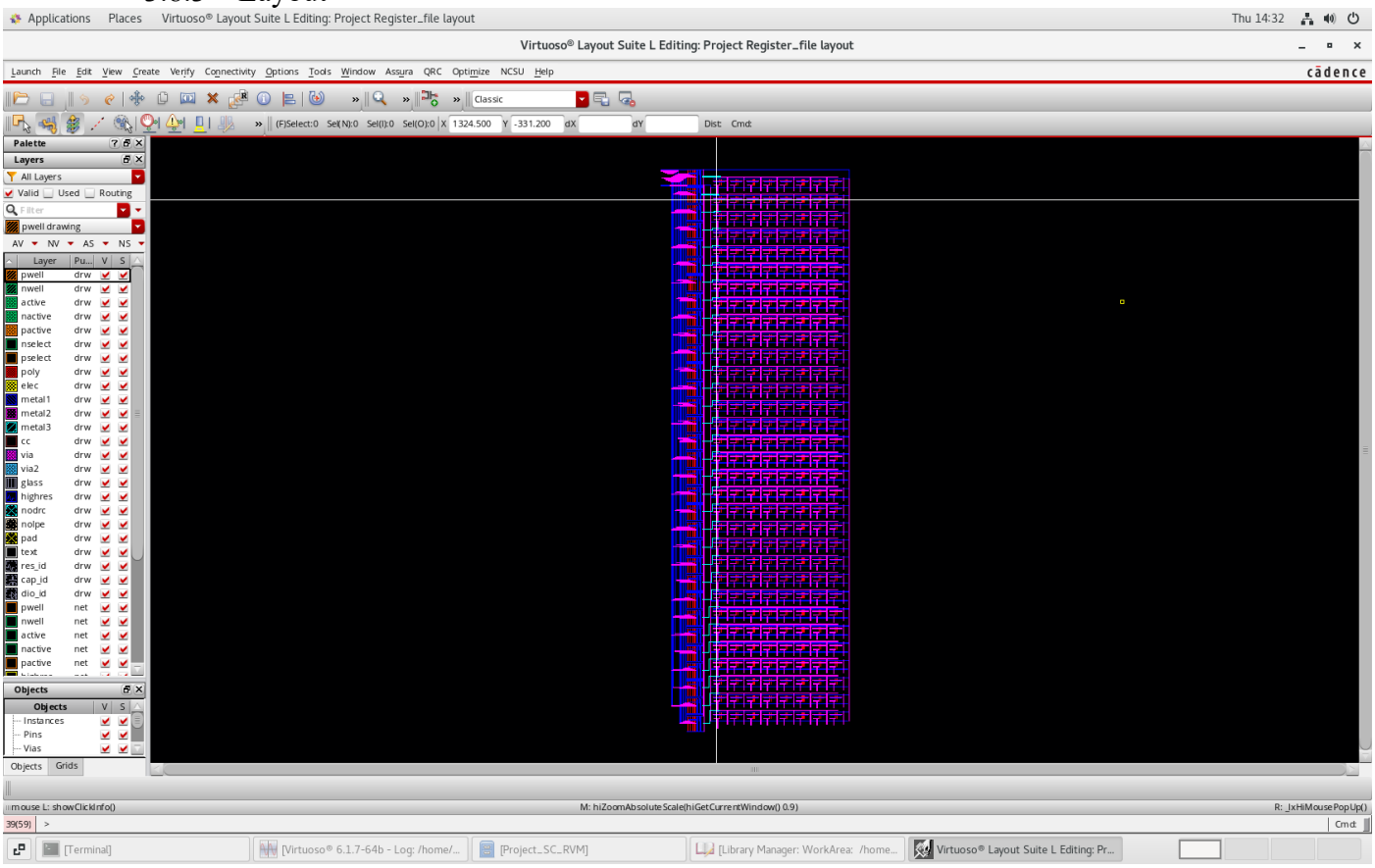
3.8.1 Schematic

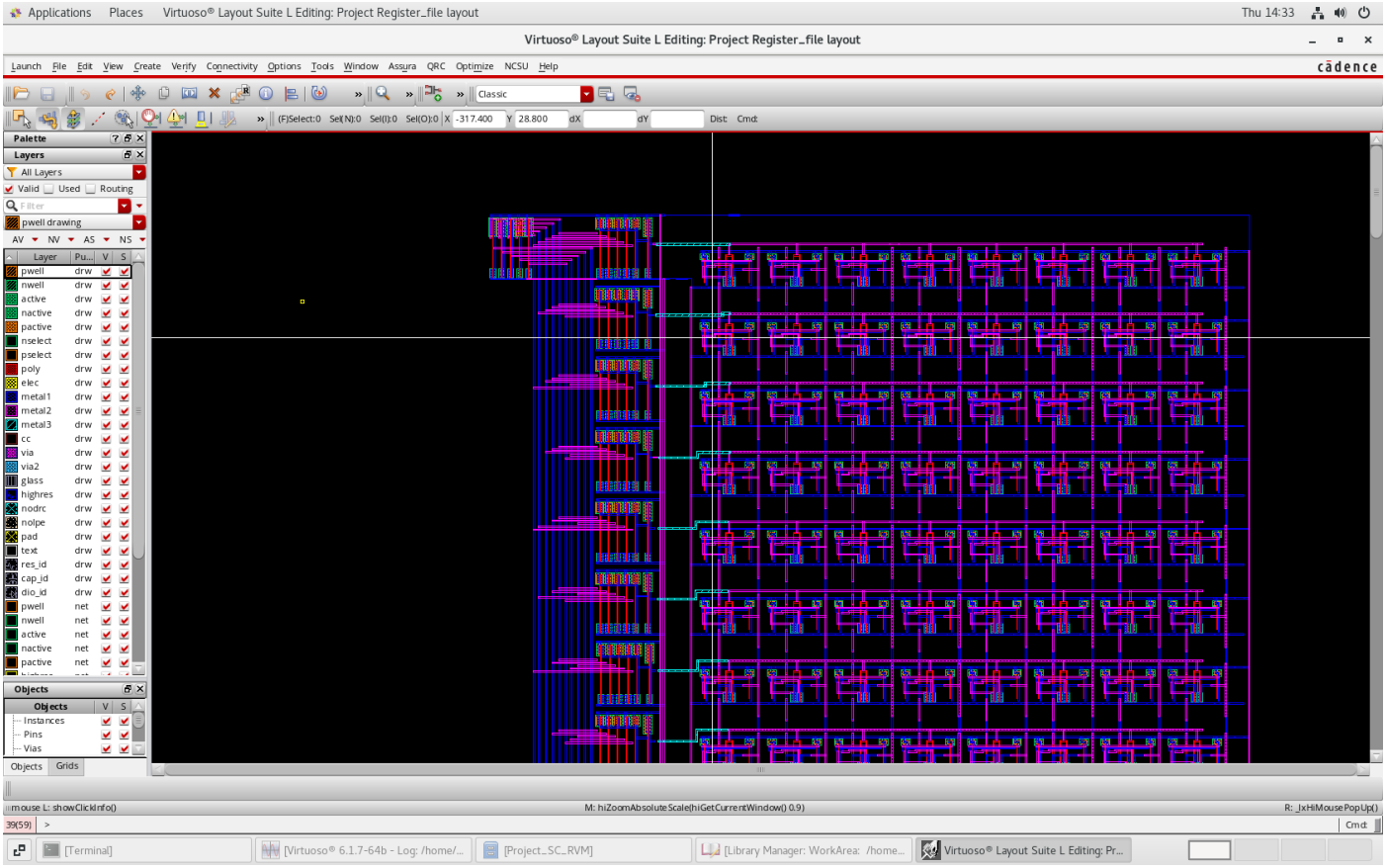


3.8.2 Symbol

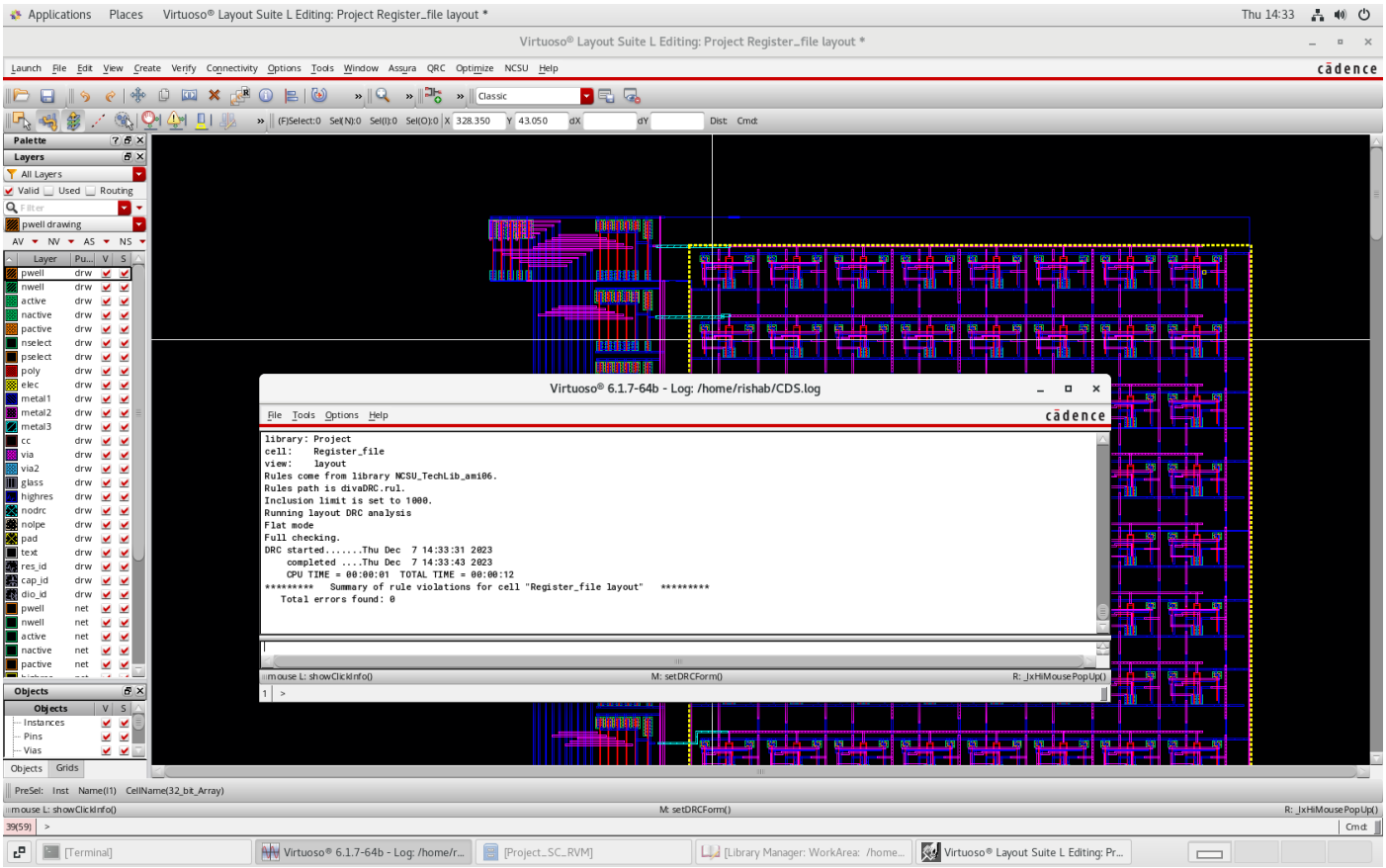


3.8.3 Layout

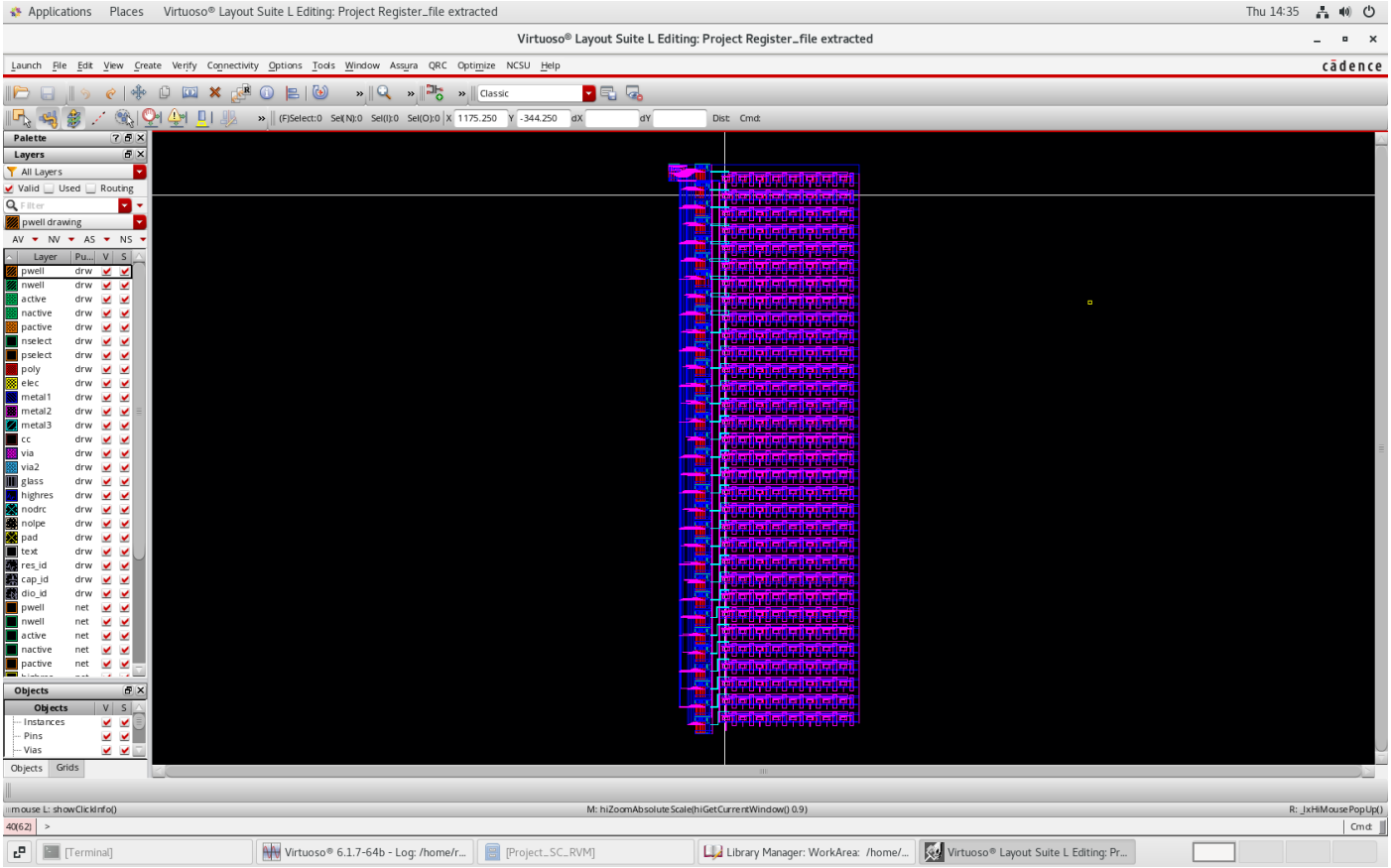

Applications Places Virtuoso® Layout Suite L Editing: Project Register_file layout
Thu 14:32
Virtuoso® Layout Suite L Editing: Project Register_file layout
cadence

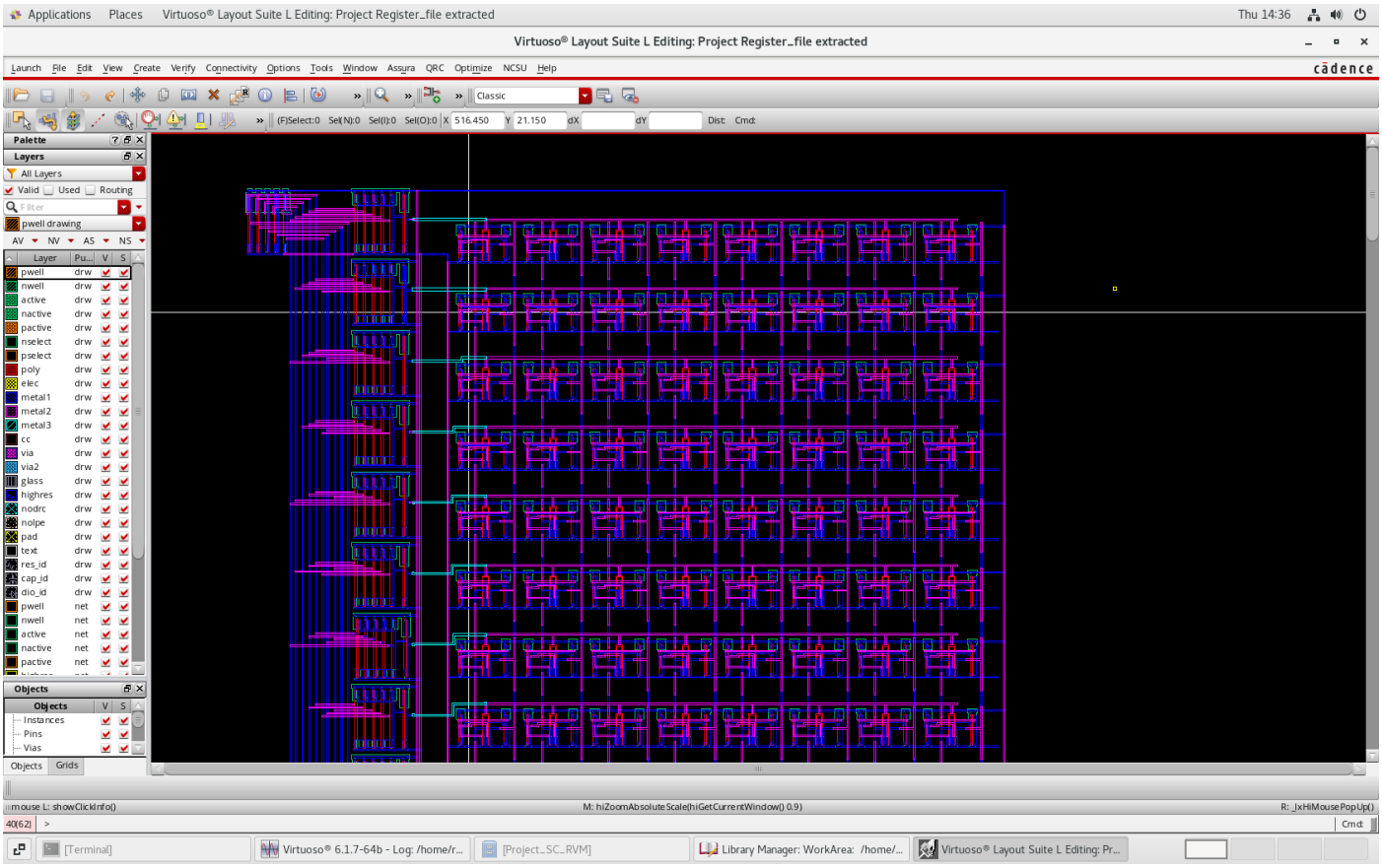

Applications Places Virtuoso® Layout Suite L Editing: Project Register_file layout
Thu 14:33
Virtuoso® Layout Suite L Editing: Project Register_file layout
cadence

3.8.4 DRC

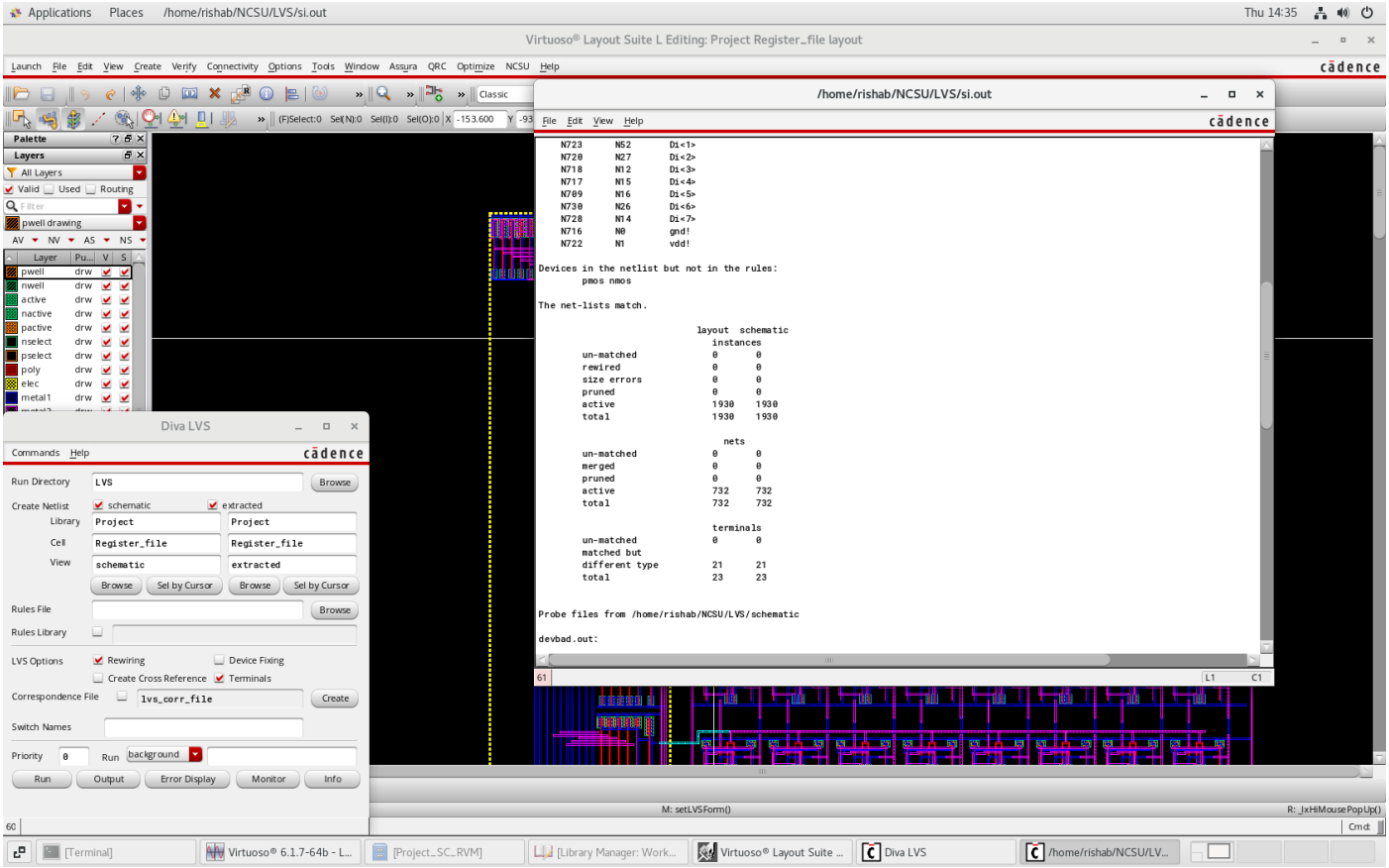


3.8.5 Extracted

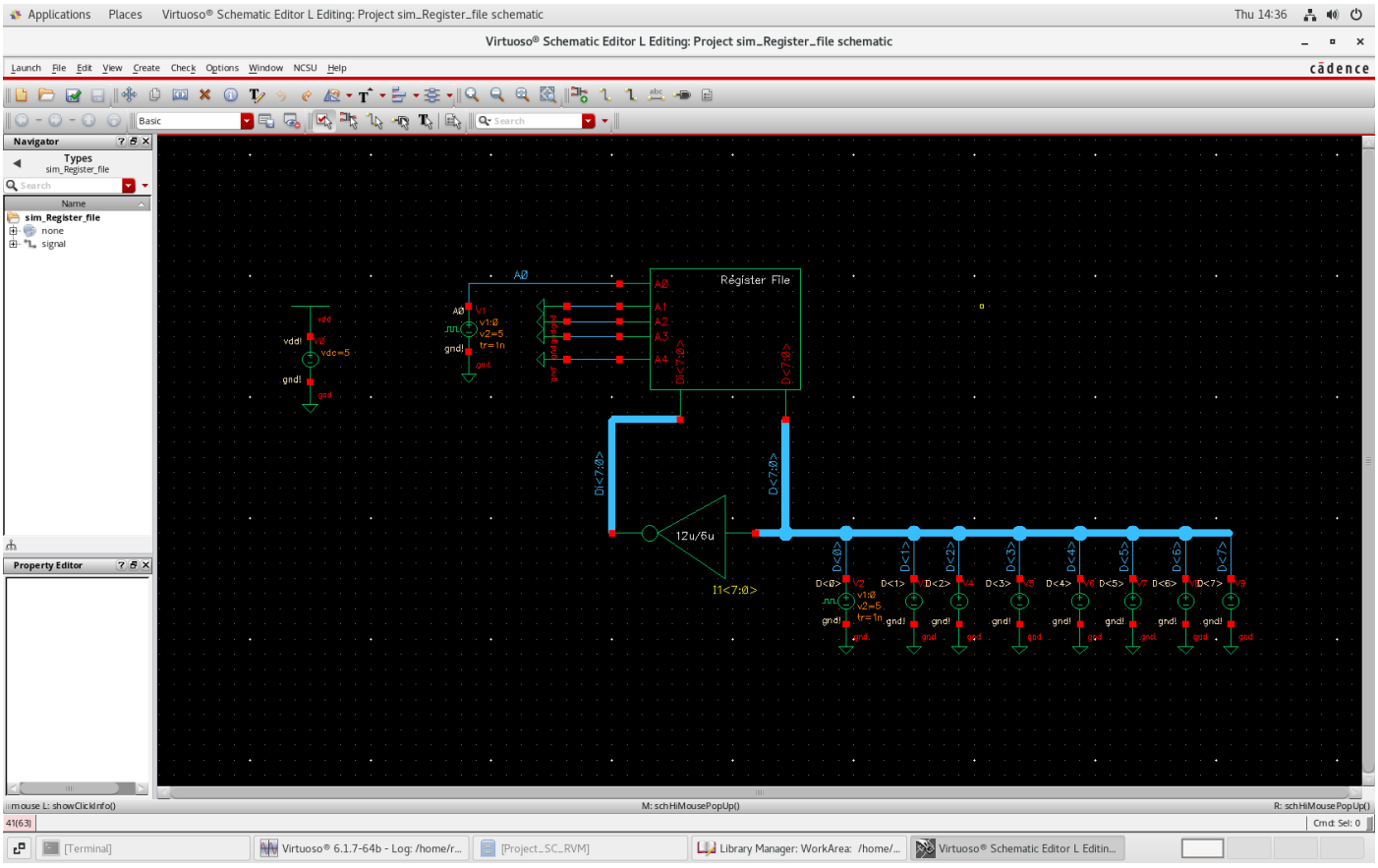


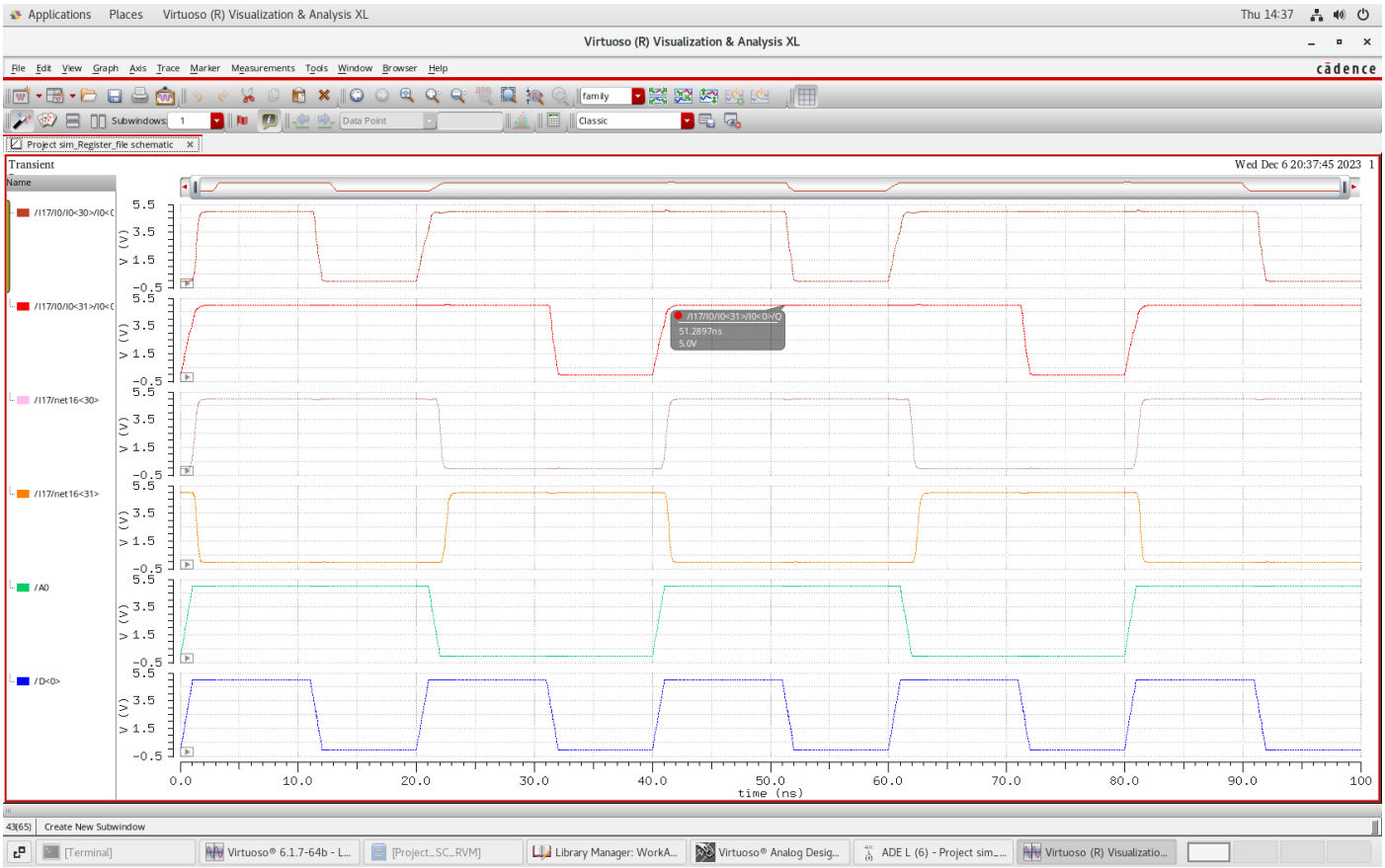

Applications Places Virtuoso® Layout Suite L Editing: Project Register_file extracted
Thu 14:36
Virtuoso® Layout Suite L Editing: Project Register_file extracted
Launch File Edit View Create Verify Connectivity Options Tools Window Assura QRC Optimize NCSU Help
cadence
Palette
Layers
All Layers
Valid Used Routing
pwell drawing
Objects
Instances
Pins
Vias
Objects Grids
Virtuoso® 6.1.7-64b - Log: /home/r...
[Project_SC_RVM]
Library Manager: WorkArea: /home/...
Virtuoso® Layout Suite L Editing: Pr...

3.8.6 LVS


Applications Places /home/rishab/NCSU/LVS/si.out
Thu 14:35
Virtuoso® Layout Suite L Editing: Project Register_file layout
/home/rishab/NCSU/LVS/si.out
N723 N52 Di<1>
N720 N27 Di<2>
N718 N12 Di<3>
N717 N15 Di<4>
N709 N16 Di<5>
N730 N26 Di<6>
N728 N14 Di<7>
N716 N0 gnd!
N722 N1 vdd!
Devices in the netlist but not in the rules:
pmos nmos
The net-lists match.
layout schematic
instances
un-matched 0 0
rewired 0 0
size errors 0 0
pruned 0 0
active 1930 1930
total 1930 1930
nets
un-matched 0 0
merged 0 0
pruned 0 0
active 732 732
total 732 732
terminals
un-matched 0 0
matched but
different type 21 21
total 23 23
Probe files from /home/rishab/NCSU/LVS/schematic
devbad.out:
Diva LVS
Commands Help
Run Directory LVS
Create Netlist schematic extracted
Library Project Project
Cell Register_file Register_file
View schematic extracted
Rules File
Rules Library
LVS Options Rewiring Device Fixing
Create Cross Reference Terminals
Correspondence File lvs_corr_file
Switch Names
Priority 0
Run background
Run Output Error Display Monitor Info

3.8.7 Simulation





As observed, the register file is working correctly. When A0 is high, addressing "00001" writes to Row_line<30>, causing I0<30> to mirror D<0>. Similarly, when A0 is low and addressing "00000," we write to Row_line<31>, and now I0<31> mirrors D<0>.

4. Optimization

In comparison to the methodology presented in the IEEE paper, my Register File introduces optimizations in both size and efficiency. While the referenced work employs an 8x8 SRAM Memory Array operating at 100 MHz, our design incorporates a larger 32x8 SRAM Memory Array also functioning at 100 MHz. This expansion in size enhances the storage capacity and overall capability of the Register File.

Additionally, the access mechanism for word selection has been upgraded for improved efficiency. While the IEEE paper employs a 3:8 decoder, our Register File utilizes a more expansive 5:32 decoder. This enhancement in the decoding scheme allows for a more granular and precise selection of words, optimizing the read and write operations within the memory array. These optimizations collectively contribute to a Register File design that not only scales in size but also ensures heightened operational efficiency at the specified clock frequency.

5. Trouble Shooting

I encountered several challenges while designing different parts of the Register File. One major issue was that I didn't leave enough space between the PMOS and NMOS of the inverter, causing problems when instantiating it in the SRAM layout. This led to issues with metal connections. After fixing this, I struggled with selecting the instantiated schematic and figuring out how to return to the top-level schematic.

In the layout design of the 32-bit Array, I faced DRC issues where I had placed the instantiated 8-bit SRAM with incorrect connections to the pins. Fixing this, I then encountered a net-list mismatch in the Layout vs Schematic check, with twice as many nets in the layout compared to the schematic. This was resolved by realizing that my SRAM words were not properly interconnected, and I corrected the connections.

Another problem occurred during the Register File layout design, where I had misconnections of the Row_line from the decoder to the Row_line of the 32-bit Array. I spent a lot of time manually drawing metal rectangles to make connections until Professor Scott shared a helpful shortcut (pressing 'p') that made the process much quicker.

Lastly, during the Register File simulation, I noticed glitches and excessive noise in the graph. After about an hour of troubleshooting, I discovered that I had flipped the wire connections in the simulation schematic (connecting D with Di pin and vice versa). Once I fixed all these issues, the Register File worked as intended.

6. Conclusion

In conclusion, the design of the Register File featuring a 32x8-bit word SRAM Memory Array with a 5-bit Address on Cadence Virtuoso 180nm Technology was a success. I ensured that all the component layouts met the design rule check (DRC) standards. Additionally, the verification of Layout vs Schematic (LVS) for each component was conducted, confirming that the layouts accurately matched the schematics and that the netlists were in sync. The comprehensive circuit analysis and simulations were carried out using the Analog Design Environment L, or ADE-L software. The successful completion of these steps underscores the reliability and functionality of the designed Register File.

7. References

[1] N. Snehith, E. S. Kumar and K. S. Rao, "Design and Analysis of 8×8 SRAM Memory Array using 45 nm Technology at 100 MHz," 2023 IEEE Devices for Integrated Circuit (DevIC), Kalyani, India, 2023, pp. 501-506, doi: 10.1109/DevIC57758.2023.10135059.

[2] C. S. Hemanth Kumar and B. S. Kariyappa, "Design and Power Analysis of 16×16 SRAM Array Employing 7T I-LSVL," 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 2018, pp. 319-322, doi: 10.1109/RTEICT42901.2018.9012414.