IMPLEMENTATION AND VERIFICATION OF pPIM ARCHITECTURE LAYOUT

by

SAVANKUMAR PRAJAPATI

A Research Paper Submitted

in

Partial Fulfillment

of the

Requirements for the Degree of

MASTER OF SCIENCE

in

Electrical Engineering

Approved by:

PROF_____

Dr. Amlan Ganguly, Graduate Research Advisor
Department Head, Department of Computer Engineering

PROF_____

Dr. Ferat Sahin
Department Head, Department of Electrical and Microelectronic Engineering

DEPARTMENT OF ELECTRICAL AND MICROELECTRONIC ENGINEERING

KATE GLEASON COLLEGE OF ENGINEERING

ROCHESTER INSTITUTE OF TECHNOLOGY

ROCHESTER, NEW YORK

December 2020

Abstract

This paper contains the layout generation of a very complex pPIM architecture which is used in very intensive data computation. The pPIM architecture is eventually a memory so the target is to fit the layout in a widely used memory block. The layout of complex and big memory requires very thorough planning of routing. To do the simulation, Cadence Virtuoso software is used in 45 nm technology. The first step is to design the schematic of various blocks like MUX, LATCH, register, etc., and check the dealy in the circuit. The layout of these blocks is also generated. Utilizing these basic blocks, a Core schematic is created. The layout of a very complex circuit is done using the auto-routing tool. After completing the routing, DRC and LVS verification are done which is very crucial for any layout. That will verify the sustainability of the layout in the given technology. In the end, a rough idea about the Cluster area is reported. This paper mainly focuses on layout techniques for complex circuits.

Declaration

I hereby declare that the research work done in the report is my own work except for the specific mention of the reference. All outcomes and results are achieved by me. This graduate paperwork is the partial fulfillment towards the Graduation for a master's in electrical engineering.

Savankumar R Prajapati

Dec,2020

Acknowledgments

I would like to take the opportunity to thank all my family members for their continuous support and belief throughout my dream career. I would also like to thank my colleagues for their constant support as well. At last, I would like to thank Dr. Amlan Ganguly for allowing me to work with him. Not only by giving time, but also providing valuable technical insight into the project work. Also special thanks to Prof. Mark Indovina for the help.

Contents

List of Figures

1. Introduction

Current technology follows Moore's law which predicts the number of transistors on a chip doubles every two years. With that, complexity and errors also increase while designing any circuit and layout for the fabrication. Tech companies have to be far ahead in research in developing the new technology.

One of the major aspects is the layout of any circuit. In IC design, every design passes through the layout process before fabrication. The proper layout will increase the performance of the circuit by reducing the delay and also reduce the area consumed which lowers the cost of fabrication and saved space.

In this paper, the layout of very complex pPIM architecture is designed from very basic blocks. Delays of various blocks are measured. Future work involves the clean layout of the cluster and different dealy measurements.

2. PIM Architecture

2.1 General functionality of PIM

Processing in memory (PIM) architecture is best suitable for implementing data-intensive applications such as Deep Neural Networks (DNN) and Convolutional Neural Networks (CNN). The proposed Lookup table (LUT) based PIM architecture aimed to accelerate the CNN/DNN methods by replacing logic-based processing with precalculated results stored inside the LUTs to perform complex computations on the DRAM memory[1].

The LUT based DRAM PIM architecture offers higher performance compared to conventional bit wise parallel PIM architecture. This architecture also avoids problems related to fabrication. PIM is the branch of non-Von Neumann architecture in which the processing elements are implemented on the memory chip itself. This method provides very low latency and very high energy efficiency.

PIMs are suitable for deep learning (DL) applications where relatively simple arithmetic computations to be performed on massive amounts of data. The PIM architecture is reprogrammable which leverages the LUTs to perform within the memory chip. LUT based PIMs are significantly faster and energy efficient compared to bit-wise logic circuit-based architectures due to the absence of switching power consumption of logic gates. pPIM is capable of doing virtual computation[1].

Transmission gate (TG) implementation of processing units allows a more compact design with a lower footprint and higher energy efficiency. Figure 1 shows a hierarchical view of the architecture. pPIM core provides programable operations on two 4-bit inputs. pPIM cluster is made

up of 9 similar cores. Each pPIM cluster acts as a complete processing element (PE) that can be programmed to perform a wide range of operations such as Multiply and Accumulate (MAC), Substitution, Comparison, Bitwise Logic Operations, etc. An array of these clusters can be utilized to implement different layers of CNN and DNNs.

2.2 Core structure

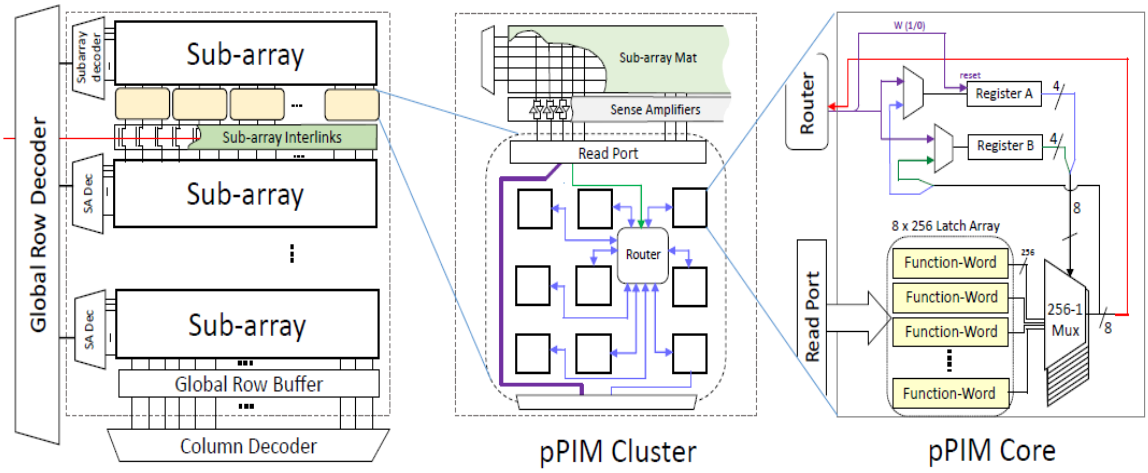

Figure 1. PIM Architecture[1]

LUT based architecture can perform multiplications with a significantly lower delay compared to bitwise operation without any tread-off in accuracy.

As shown in figure 1, LUTs are implemented with 8 - bit 256-to-1 multiplexer. The MUX select lines are controlled by two 4-bit inputs registers A and B. The input to the MUX is called function words, which are directly read from 8 arrays of latches each of which contains 256 latches. The core can be programmed by simply adding new function words into the array of latches and forwarding those as input to the MUX.

Function words are stored in the latches which improve energy efficiency as well as minimize on-chip overhead. Data inputs to the core are routed through the router of the pPIM cluster.

2.3 Cluster Architecture

9 cores are arranged together to make a cluster in 3×3 format to scale up the size of the operand. This cluster forms a 2-D array across a DRAM bank as shown in figure 4. Data communication among the pPIM cores within a cluster is achieved through a routing mechanism that is capable of establishing parallel and direct connections among all the cores. The routing mechanism allows each data communication among the cores to be split into upper and lower 4-bit segments each of which can be dispatched individually[1].

3. General Definition

3.1 45 nm PDK details and design rules

Process design kit (PDK) is the manual which is defined for every technology. This manual consists of design rules to be followed when doing the layout. Each technology has a different PDK manual which has specific rules to be followed like spacing between similar metals, the spacing between different metals, enclosures, overlaps, etc. This PDK tool is useful while solving the Design rule check error.

3.2 Design rule check (DRC)

DRC is the first step towards the physical verification of the design. Different technology has different DRC rules defined in the PDK kit. DRC rules check for the number of parameters to get the best design for the technology. It helps in avoiding the manufacturing error. Mainly three rules are checked like Enclosure, spacing, and width as shown in figure 2[3].

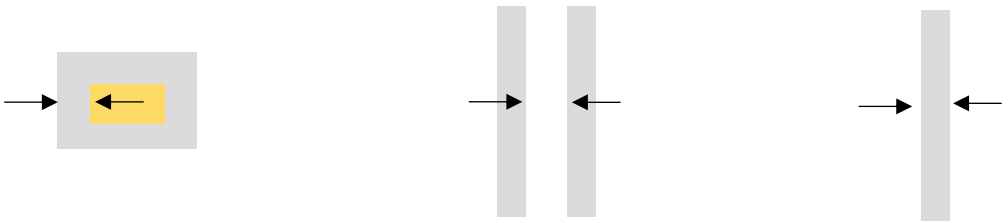

Figure 2. Enclosure, Spacing, Width

3.3 Layout verse schematic (LVS)

After solving all the DRC errors, the next step is to check for the LVS errors. DRC does not check the mismatch between the schematic and layout. So it becomes vital to compare schematic and layout. Sometimes layout passes through DRC but it still has some overlapping

which makes shorts in the system. General LVS errors are mismatches, shorts, missing nets. Always check for VDD and VSS connection before solving LVS errors because most of the errors generated in LVS are due to the bad connection of VDD and VSS.

4. Core Area calculation

- In 28 nm technology, DRAM size is 60 mm^2.
- Each DRAM has 8 banks.
- 1 Bank area = 7.5 mm^2.
- Also, each bank has 32 subarrays as shown in figure 3.
- So Subarray area = 234375 μm^2

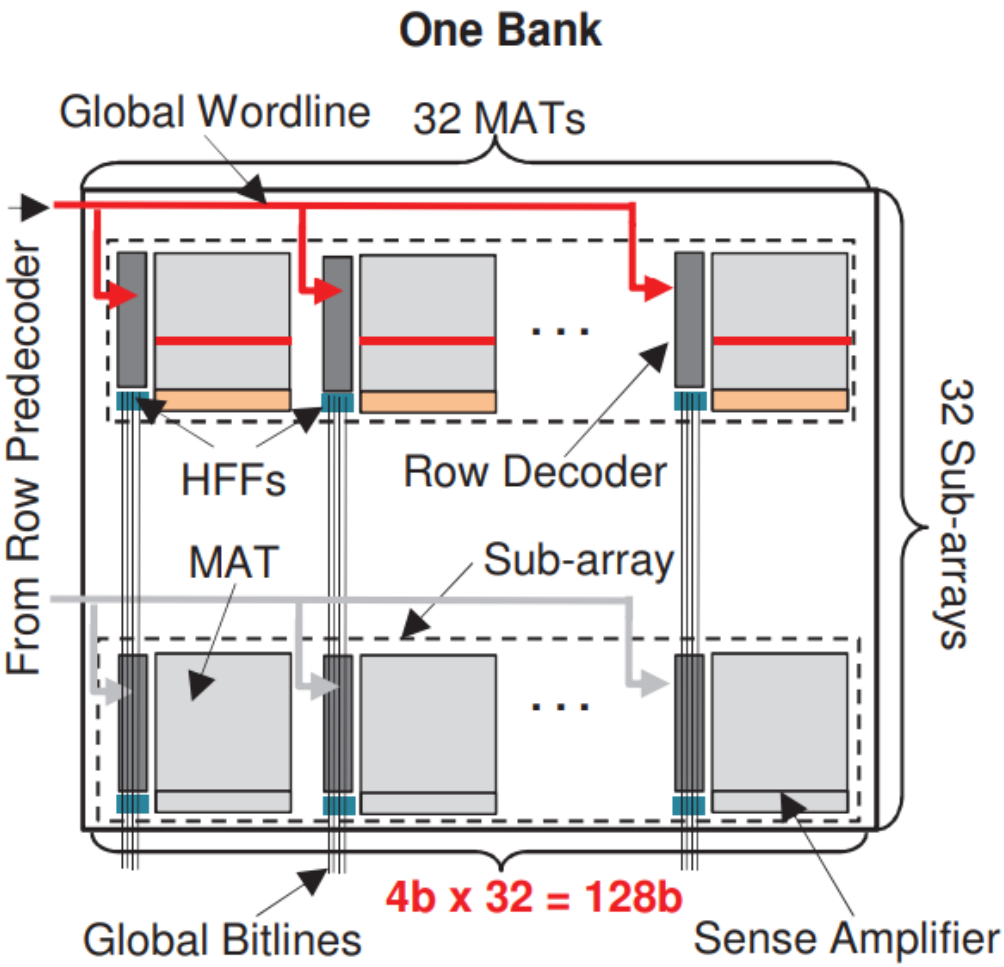

Figure 3. One bank structure in DRAM[2]

- Bank Architecture has 32 sub-arrays and each sub-array consists of 32 MATs.
- MAT is a predefined block in the DRAM which is square in dimension.
- From the subarray area, MAT area and dimensions can find out as shown below.
    - 1 MAT = 7324.22 μm^2 , L = 85.58 µm
    - 1 cluster = 4 MATS = 29296.88 μm^2
    - 1 cluster = 342.32 * 85.58 μm^2 (in 28 nm)
    - 1 cluster = 24451.42 * 6112.85 λ^2
    - 1 cluster = 611.28 * 152.82 μm^2 (in 45 nm)
- Note : 1 λ = 0.014 µm & 0.025 µm (In 28 nm & 45 nm technology respectively)

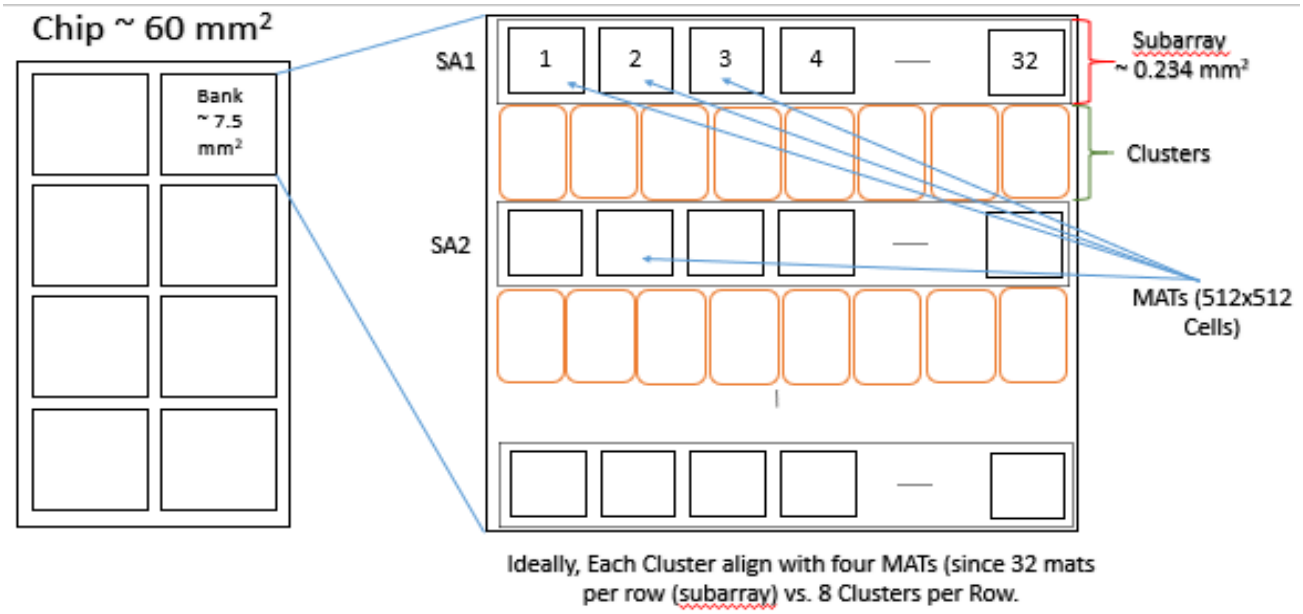

Figure 4. Subarray Structure and Cluster

- 1 Cluster is made of 9 core block. The best way to fit the cores with given dimensions is by using a 3×3 matrix form in the layout.

- 1 Cluster area = 9 core area so 1 Core area = 50.94 * 203.76 μm^2

Achieving this area is one of the core aspects of the paper.

5. Basic blocks to make a core structure

As shown in figure 1, a core design consists of two 4-bit registers, eight routers (8*1 MUX), and 8 LUTs. The register is designed using 2 latches for 1-bit so, for the 4-bit register, 8 latches are used. 8*1 MUX is designed using transmission gates which consume less area at the same time also provide the strong 0 and 1. LUT is made of 256 MUX and 256 latches. The latch will store the word and pass it to the mux for operation.

This section contains a schematic, symbol, and layout for each basic block used in making the core layout.

5.1.1 MUX schematic

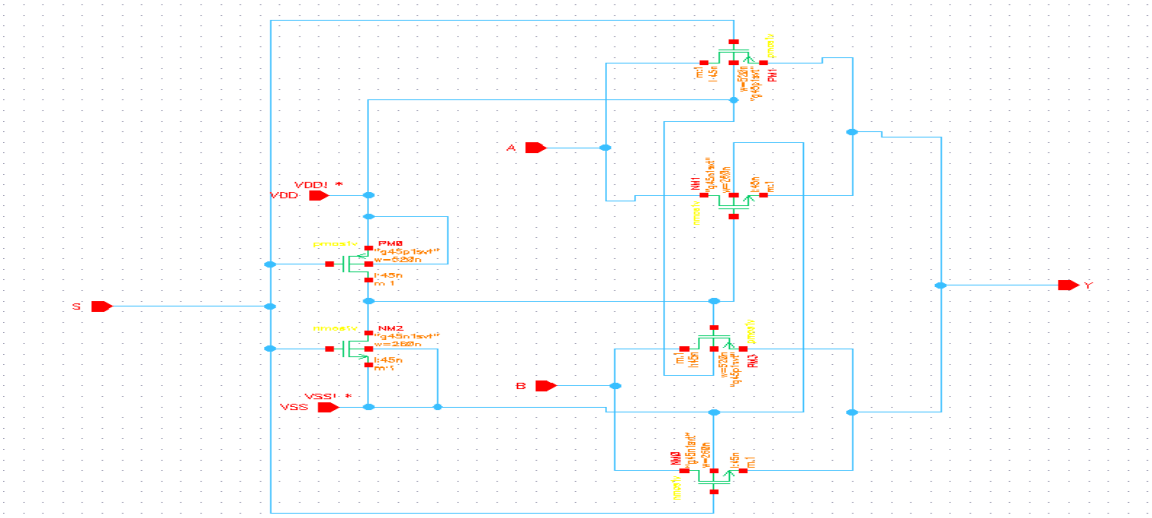

Figure 5. MUX schematic

- NMOS ratio : 280nm/45nm & PMOS ratio : 520nm/45nm

5.1.2 MUX symbol and layout

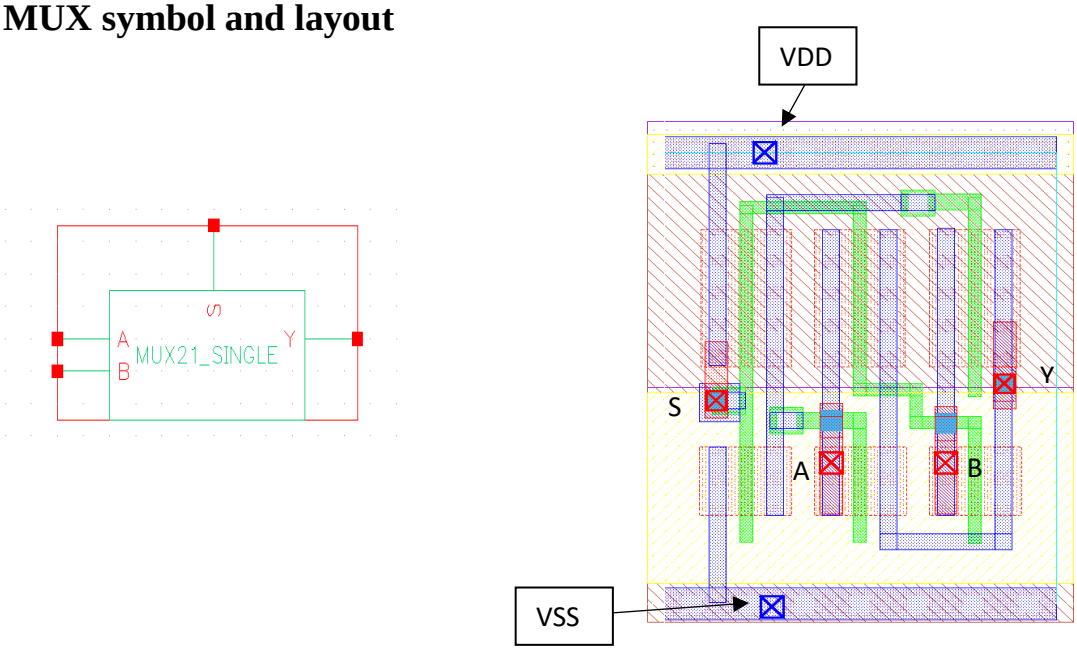

Figure 6. MUX symbol and layout

Area	1.4*1.71 μm^2
Rising Propagation delay	357.3p
Falling Propagation delay	423.7p
Avg delay	390.5p

5.2.1 LATCH schematic

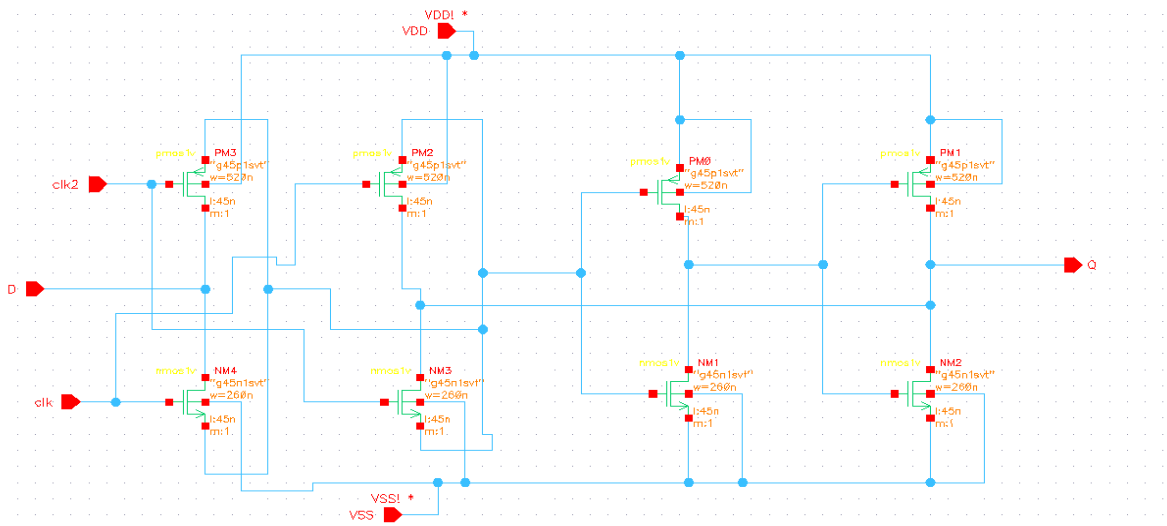

Figure 7. Latch schematic

5.2.2 LATCH symbol and layout

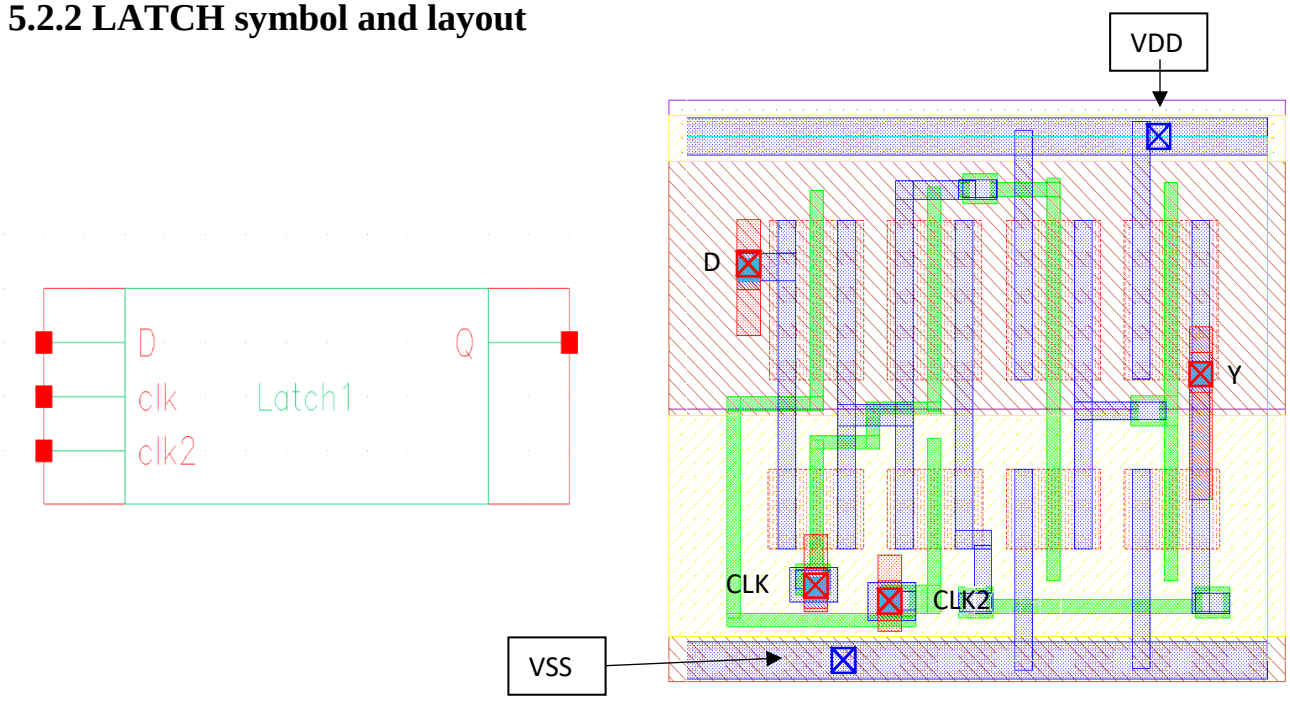

Figure 8. LATCH symbol and layout

Area	2*1.71 μm^2
Rising Propagation delay	330.97p
Falling Propagation delay	531.56p
Avg delay	431.27p

5.3.1 1-bit register schematic

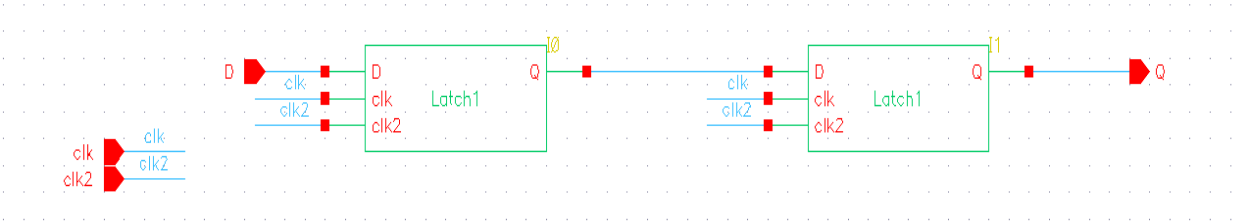

Figure 9. 1-Bit register schematic

5.3.2 4-Bit register schematic

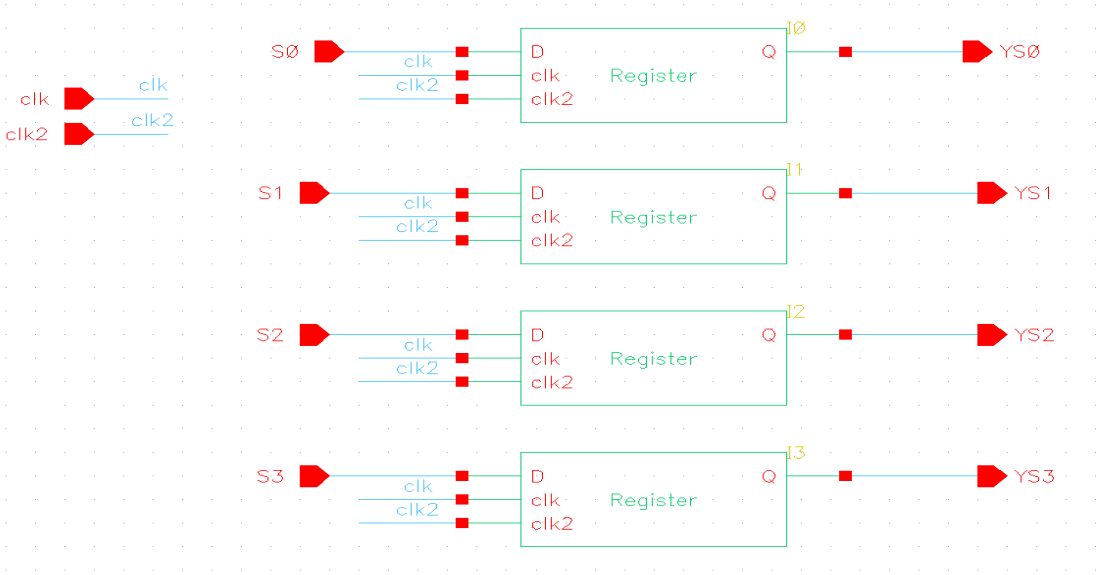

Figure 10. 4-bit register schematic

5.3.3 4-Bit register symbol and layout

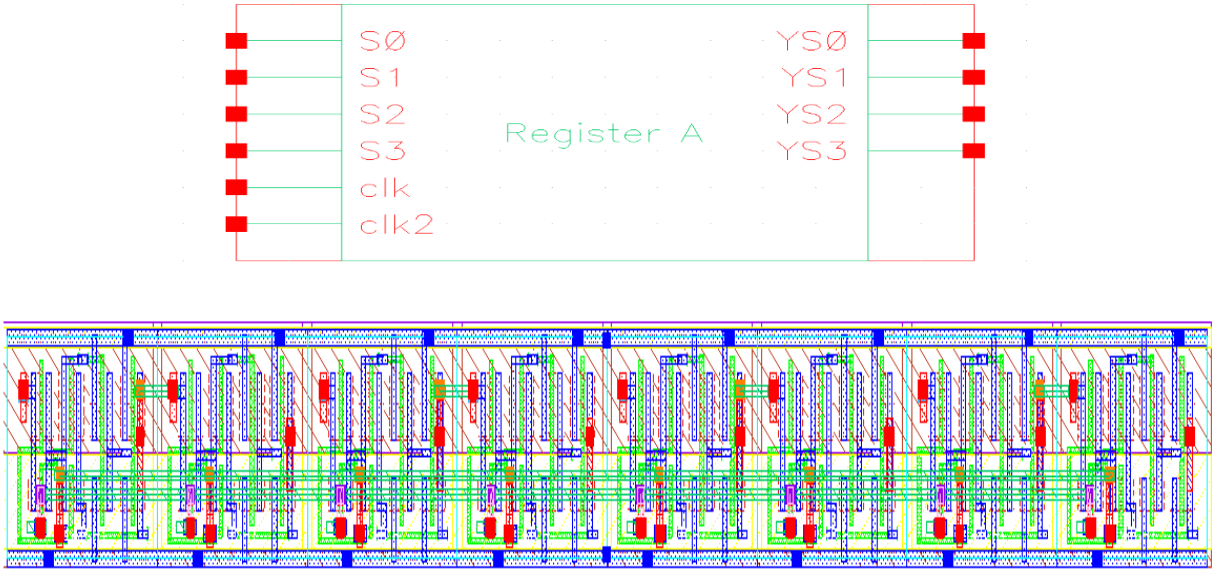

Figure 11. 4 Bit register symbol and schematics

Area	16*1.71 μm^2

5.4.1 Single TX gate schematic

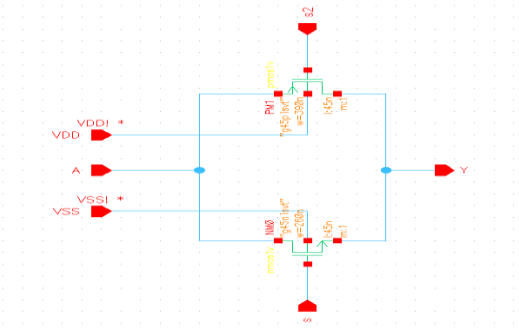

Figure 12. Transmission gate schematic

- NMOS parameter : 260nm/45nm & PMOS parameter : 390nm/45nm

5.4.2 8*1 MUX schematic

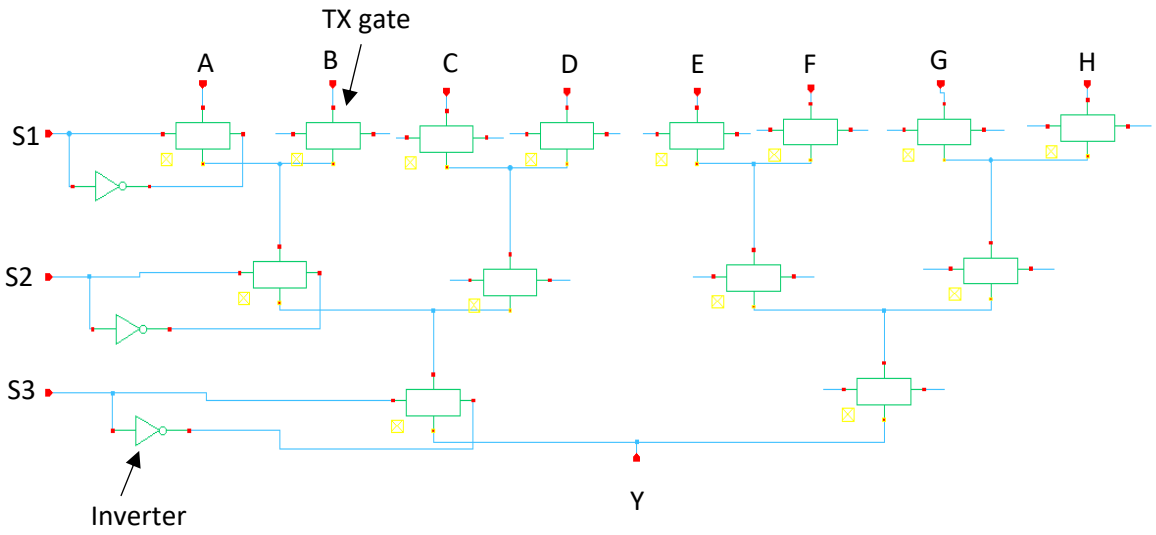

Figure 13. 8*1 MUX schematic

5.4.3 8*1 MUX symbol and layout

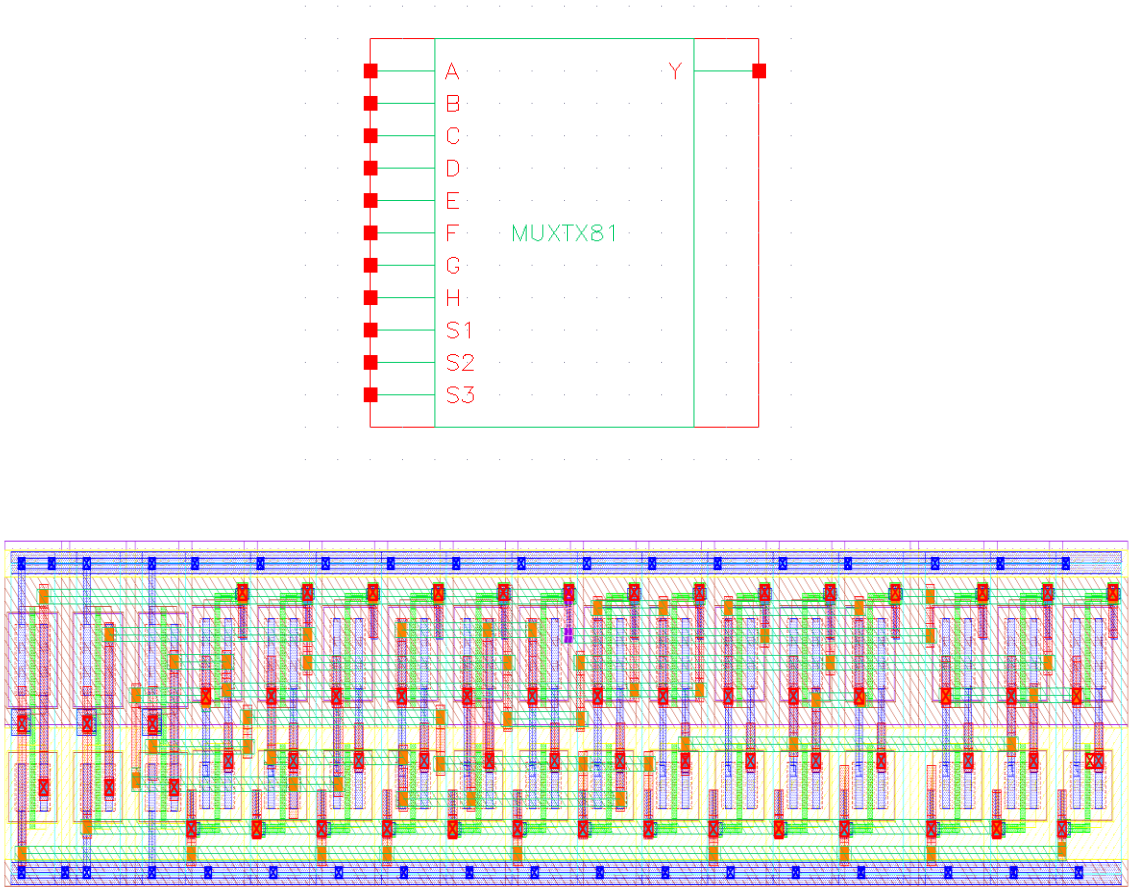

Figure 14. 8*1 MUX symbol and layout

Area	10.2*1.71 μm^2

6. Autoroute Layout steps

Cadence virtuoso layout suite GXL is used to do the layout using a 45 nm process design kit (PDK). Various steps are followed to achieve block-level design called schematic followed by layout generation from source, placement planning, automatic placement, pin optimization, automatic routing, design rule checks, and parasitic extraction.

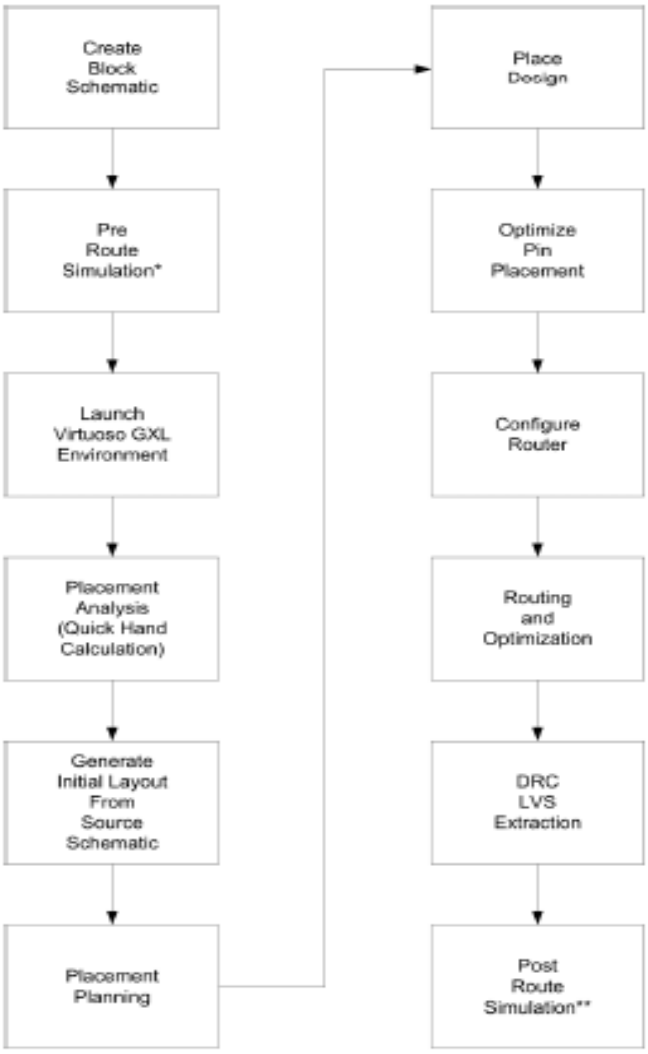

Figure 15. Layout and verification steps [4]

Figure 15 provides place and route design flow to be followed to design any layout. There are simple 8 steps to be followed to do the routing which is explained below.

6.1 Schematic

The first step is to complete the schematic as shown in figure 16 for a 256 MUX+LATCH. The top portion of the schematic contains the LATCH and the bottom portion contains MUX.

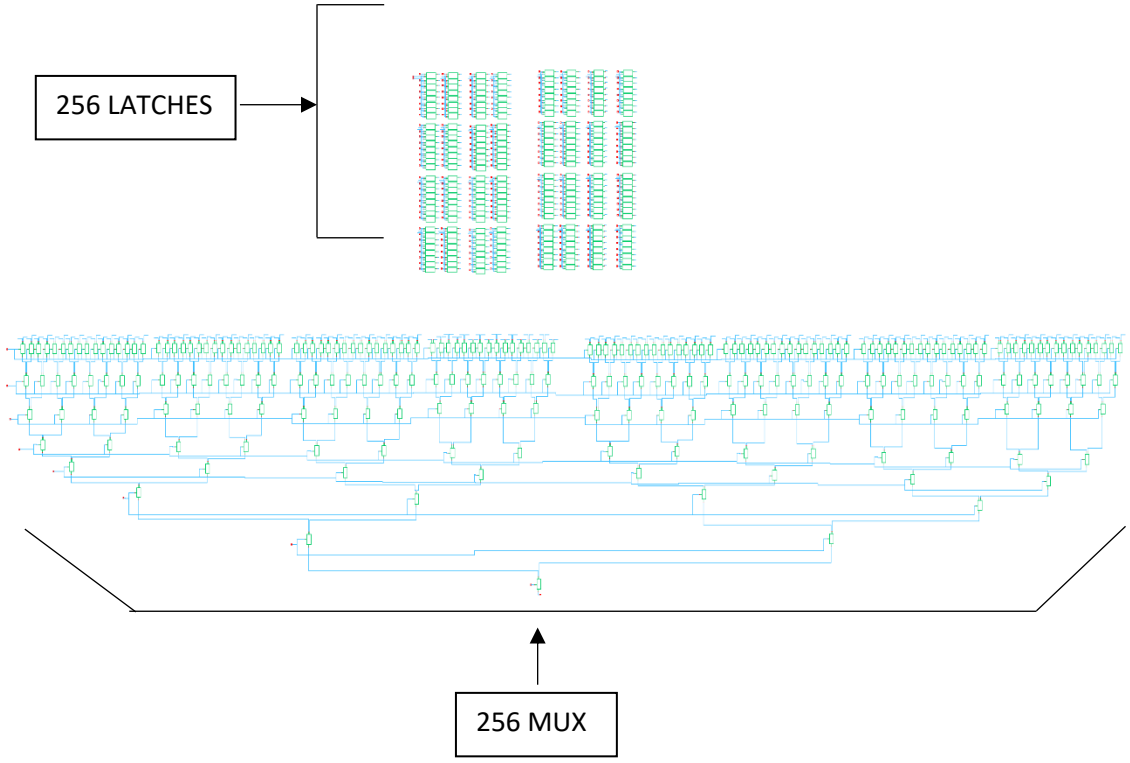

Figure 16. LUT schematics

6.2 Hierarchal block

From the schematic window, select Launch > Layout GXL as shown in figure 17. This will lead to the physical implementation startup options window to create the layout view. In the layout

window, Select Launch > Configure Physical Hierarchy and select Components types as shown in figure 18.

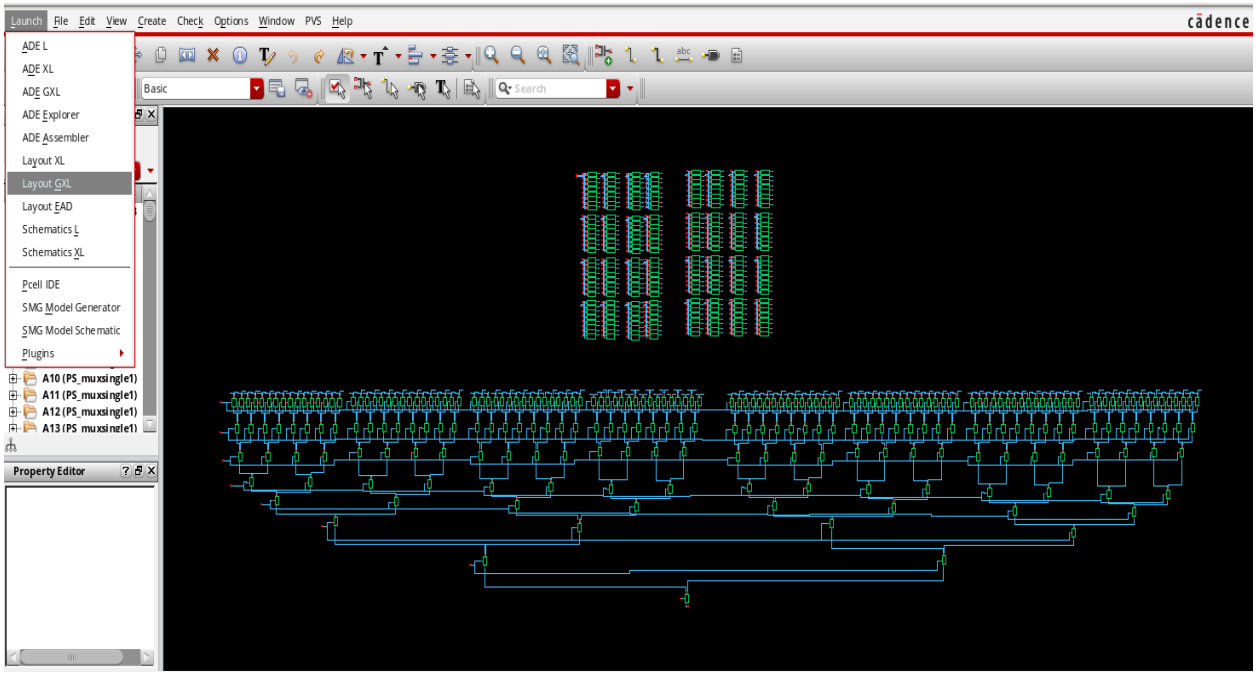

Figure 17. Launch Layout GXL

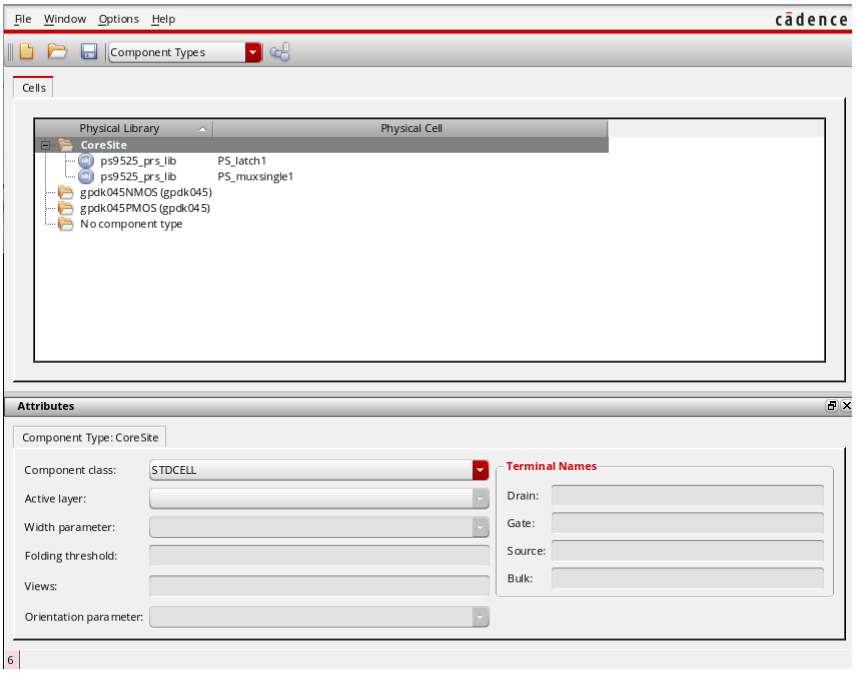

Figure 18. Config Hierarchy

If CoreSite is not created already, right-click on the open window and add component type naming CoreSite and move all the basic blocks into the CoreSite as shown in figure 18.

In the core layout problem, CoreSite contains basic blocks of PS_latch1 and PS_muxsingle1. Also, make sure to select STDCELL at the component class and save the window. This process will set up the hierarchy for the core design.

6.3 Placement Strategy

Place and route method required thorough planning before doing any placement. The goal is to create the final block layout as shown in figure 19.

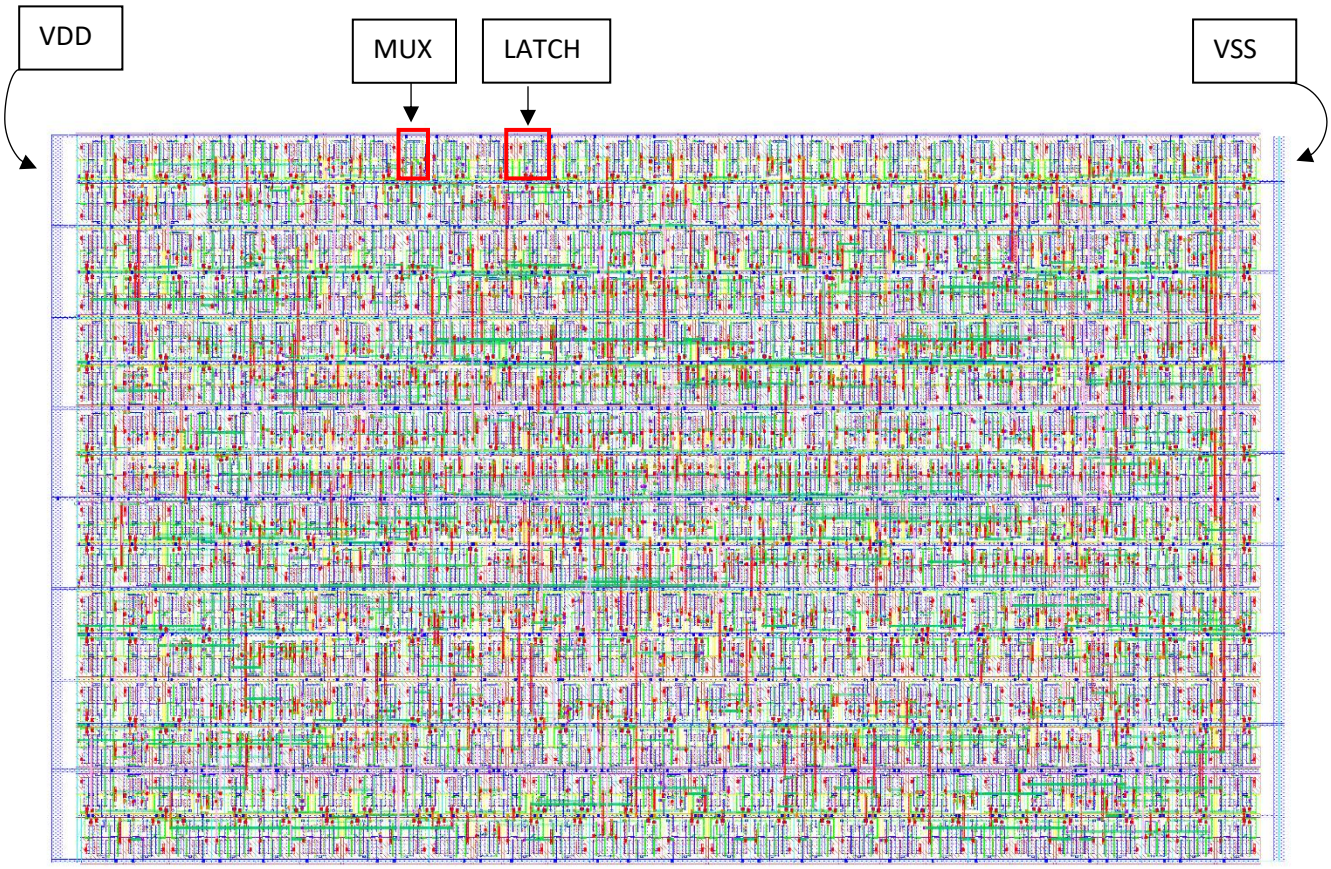

Figure 19. LUT layout

First, do a quick calculation of the area using basic block dimensions on paper. For layout related to memory, the major concern is the area so no routing tracks between the rows. For a generic 45 nm process, 1 λ = 25 nm.

In general, input and output pins are placed on the top and bottom respectively. Power rails VDD and VSS are placed on the left and right sides respectively. The width of the power rails should be at least 30 λ but due to the area constraint, 15 λ is provided.

6.4 Generate layout from source

There are different ways to generate the layout from the source. One way is by selecting one by one blocks from the schematic and generating the corresponding layout. The second way is to generate all the layouts for all blocks at a time. At the current stage looking at the complexity of the system, the second way is the way to go.

In the layout window, select Connectivity > Generate > All From Source. Gen From Source automatically creates layout objects for the various components of the schematic. It also sets the layout boundary which is called Place and Route Boundary (PR Boundary). Routing of the wiring cannot extend beyond the PR Boundary.

Make the changes in the Generate tab as shown in figure 20. Click on the I/O pins tab. Select Pin labels and click options. Make necessary changes as shown in figure 20.

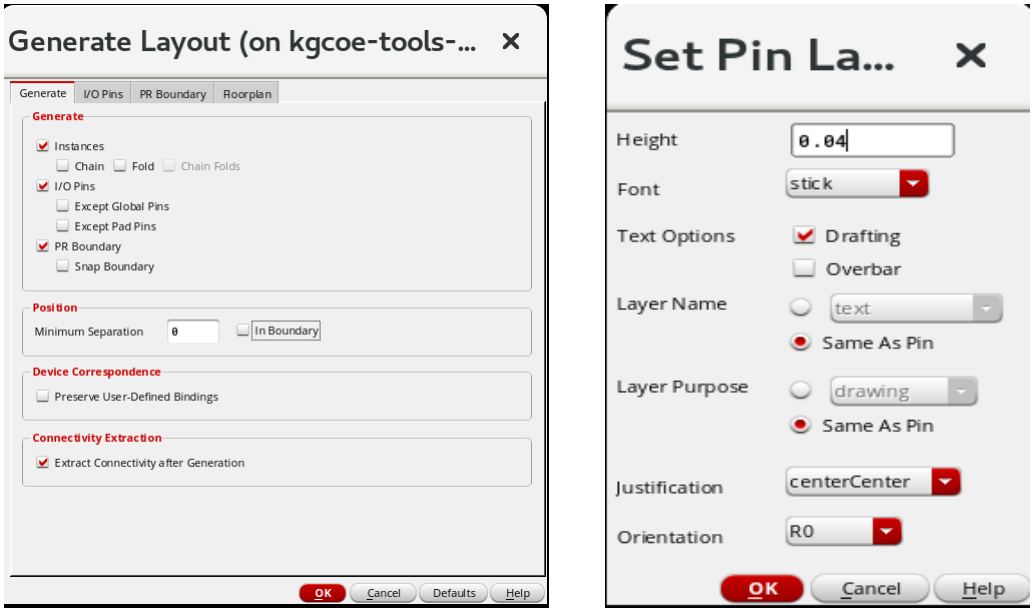

Figure 20. Generate from source - Generate

In the I/O tab, make the VDD, VSS to metal1 pin, and for input & output pins, change to Metal2 Pin as shown in figure 21.

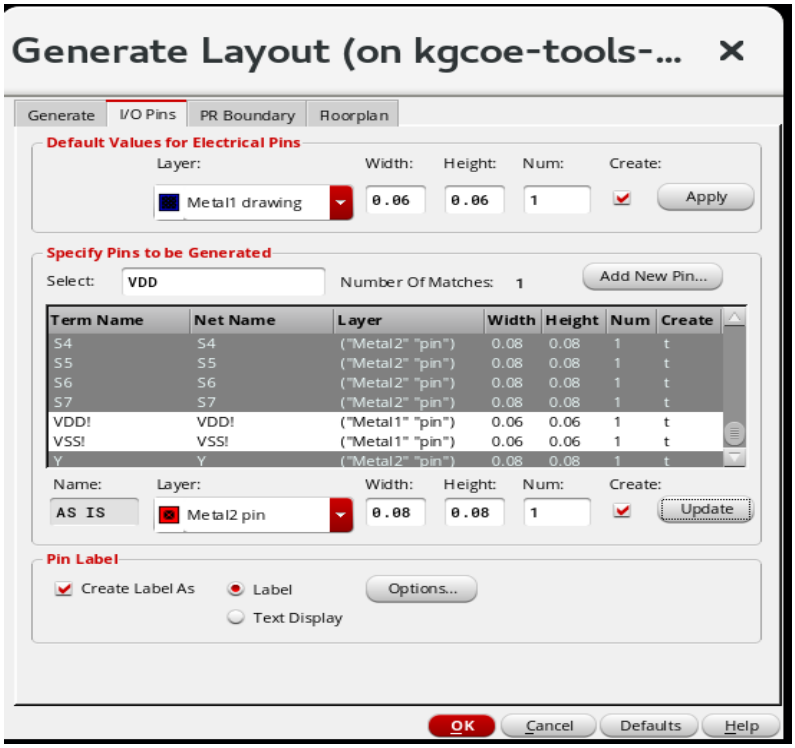

Figure 21. Generate from source - I/O pins

The next step is to define the PR boundary region as shown in figure 21.

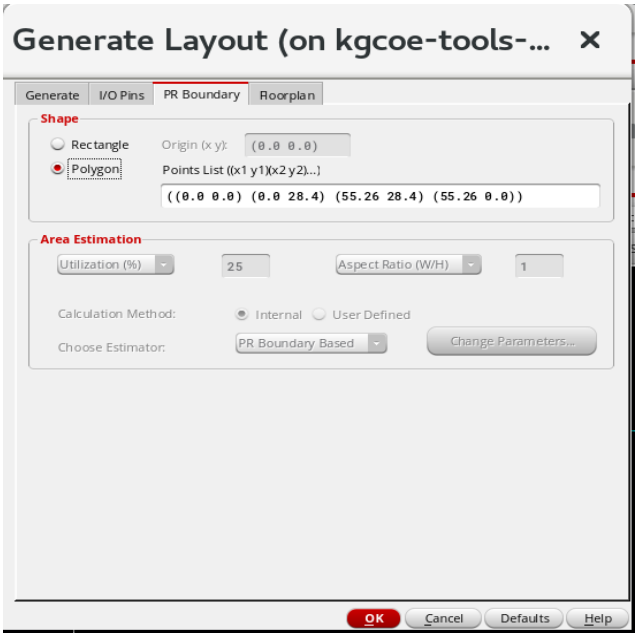

Figure 22. Generate from source -PR boundary

After defining the PR region values, click OK that generates figure 23.

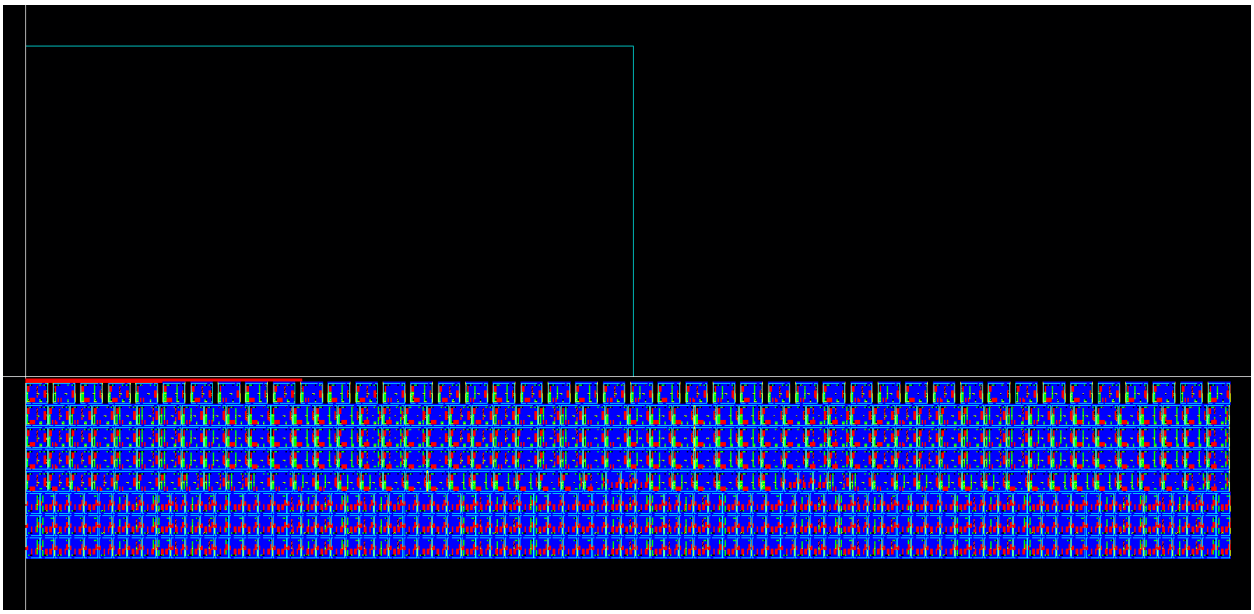

Figure 23. Layout from all generate with PR region

The blue rectangle is the PR region and the other are the MUX and LATCH blocks. At this stage, check the pins naming. This is the best time to check and verify. If any pins are not matching make necessary changes or since there is not much work done so it will be a good approach to follow the previous steps.

6.5 Placement planning

Mainly 3 steps to be followed for placement planning.

1. Previous work done for the PR boundary will make sure no routing goes out of the boundary. This PR region will be for the blocks. No blocks can be placed outside this boundary.

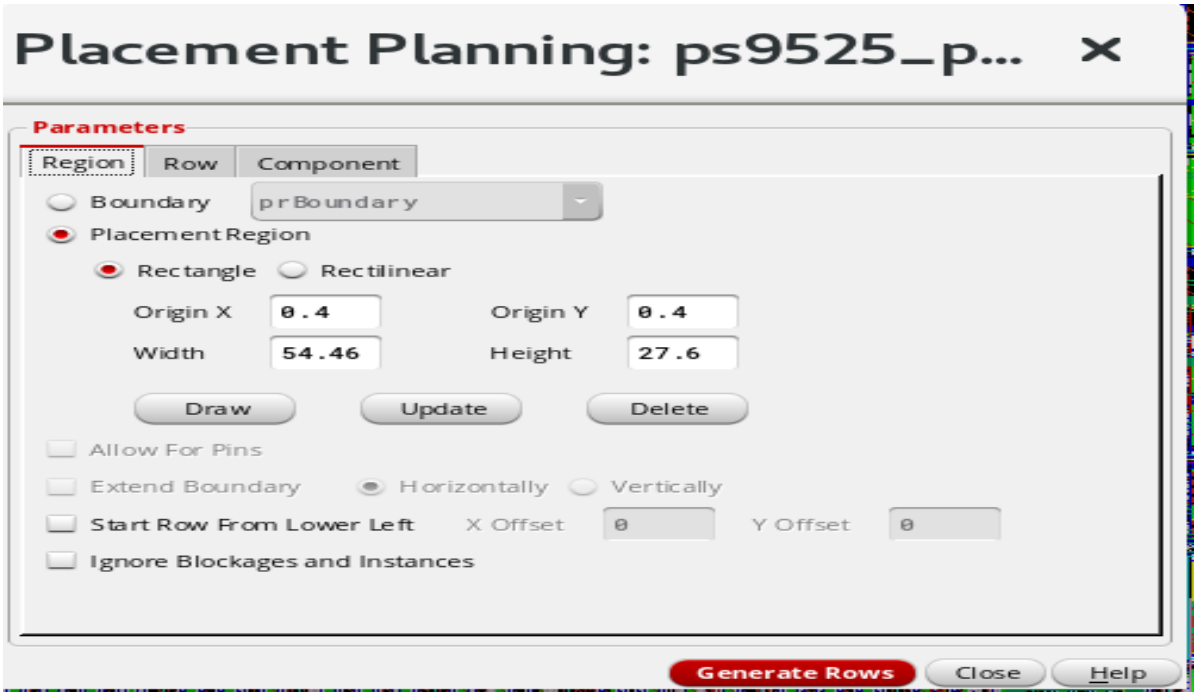

Figure 24. Placement planning - Region

Select Place > Custom Digital > Placement planning. Assign the width and height in figure 24. Next, select the Row tab and make necessary changes as shown in figure 25.

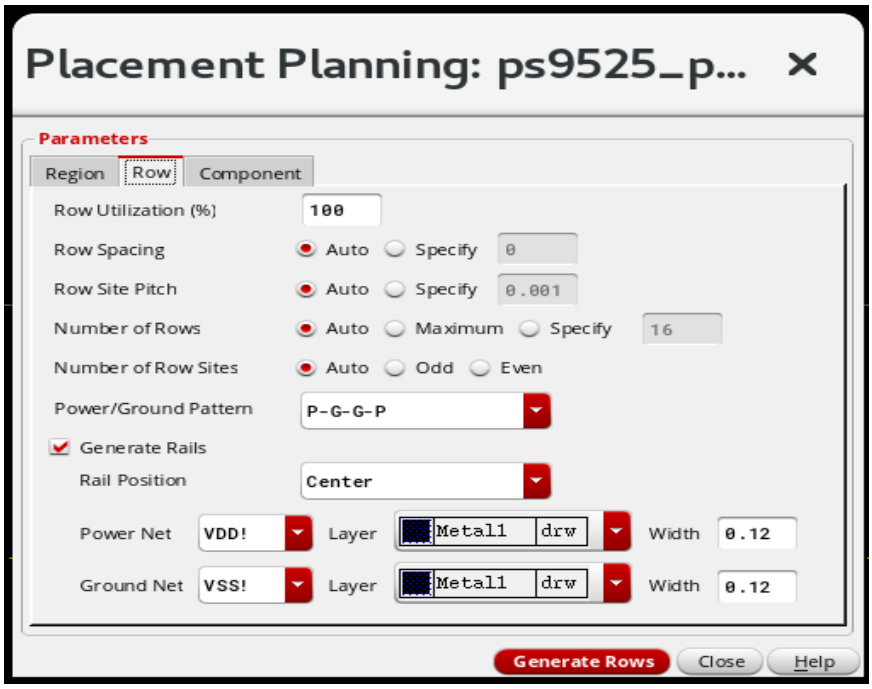

Figure 25. Placement planning - Row

The target is to make 100% packing. From the hand calculation, figure out that around 16 Rows will be needed and a good approach in combining the consecutive power rails since it will save the area. So the P-G-G-P option is selected. After doing all the changes as shown in the figure, Click on Generate Rows.

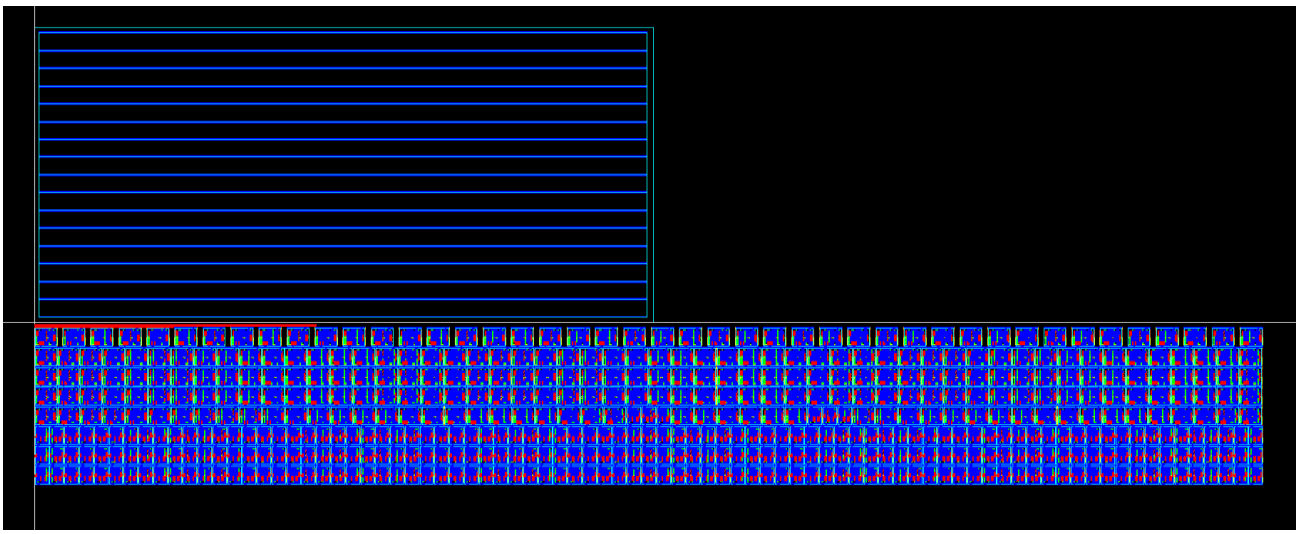

Figure 26. Layout with generate rows

Once rows are generated, the CIW window will show 100% utilization.

Note: In some of the layouts, it is impossible to achieve 100% packing so try to get as much as the packing. In this layout, 94% of utilization is made.

2. Place > Custom Digital > Dressing template

These changes will make sure the rail positions as well as the orientation on the basic blocks in the layout.

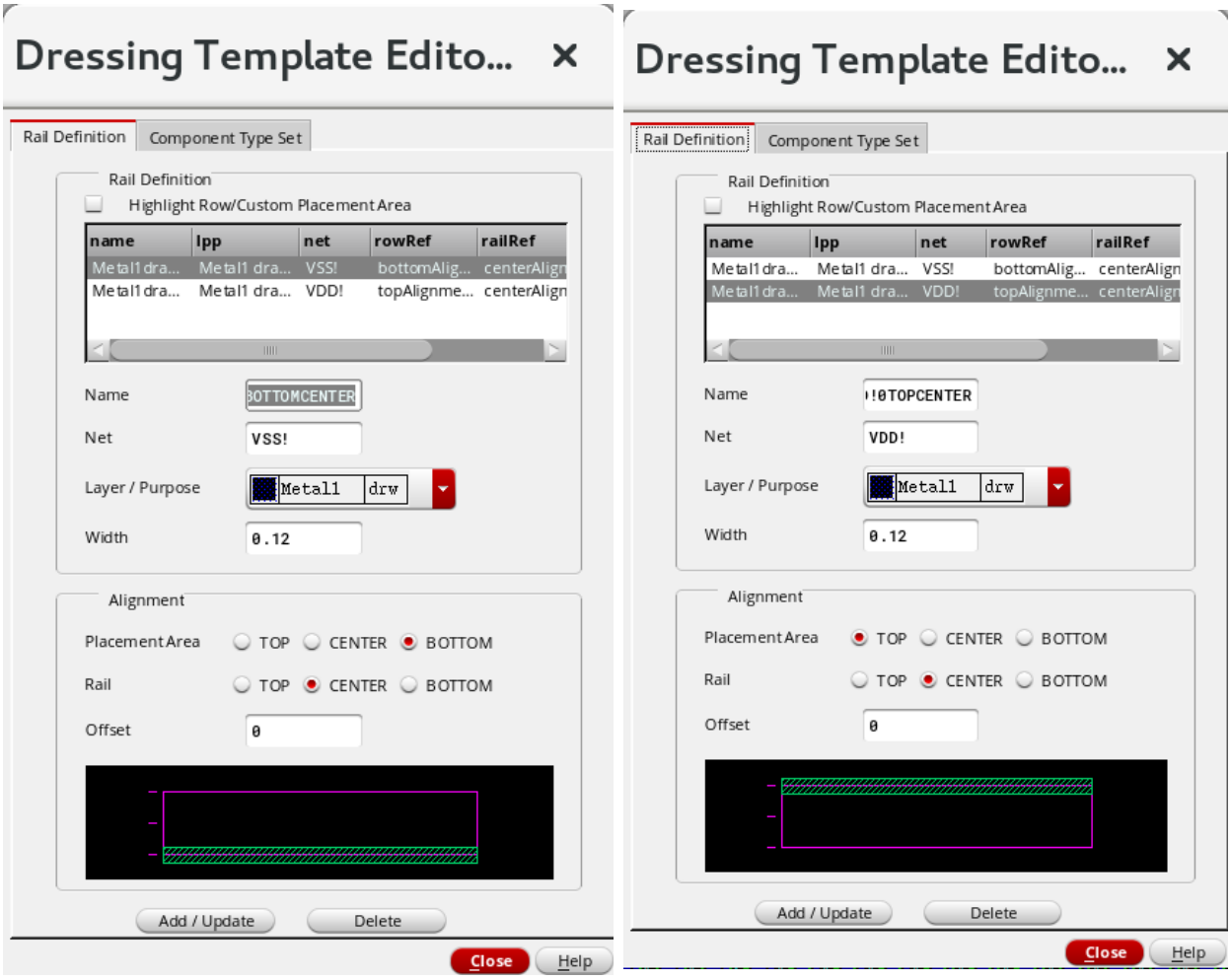

Figure 27. Design Template - Rail Definition

Make necessary changes as shown in figure 27 for the VDD and VSS Nets. For the component type set, make the changes as shown in figure 28. Sometimes orientation needed to be changed as per the layout requirement. Over here MX and MY are enabled for that purpose. After doing changes, click on Add/Update option and close the window.

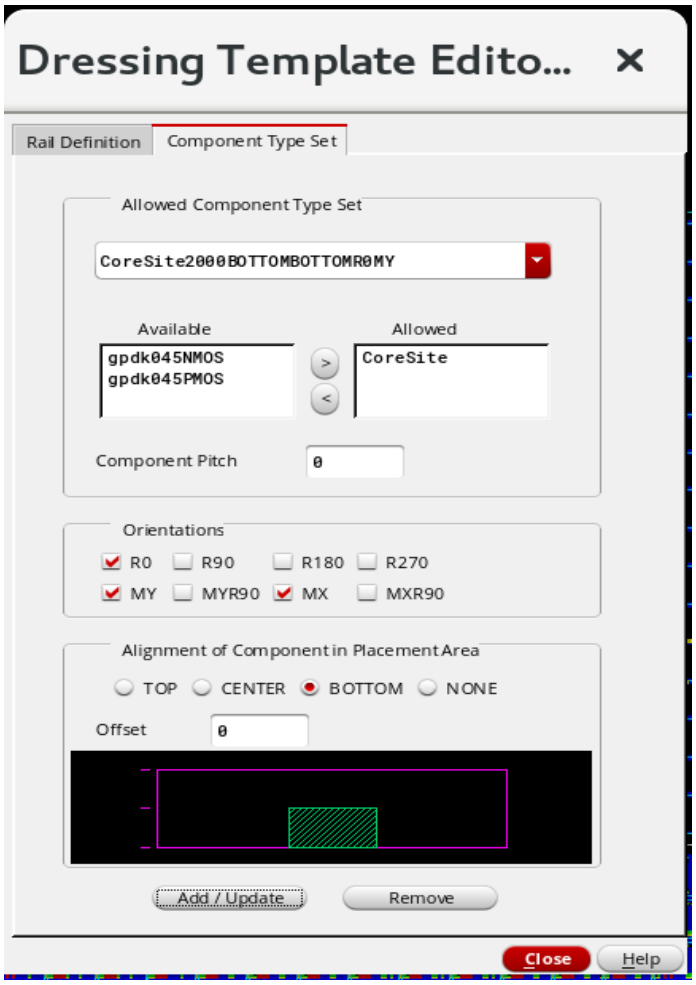

Figure 28. Design Template - Component Type Set

3. Place > Custom Digital > Auto Placement

Make necessary changes as shown in figure 29 for the Auto placer window and click OK.

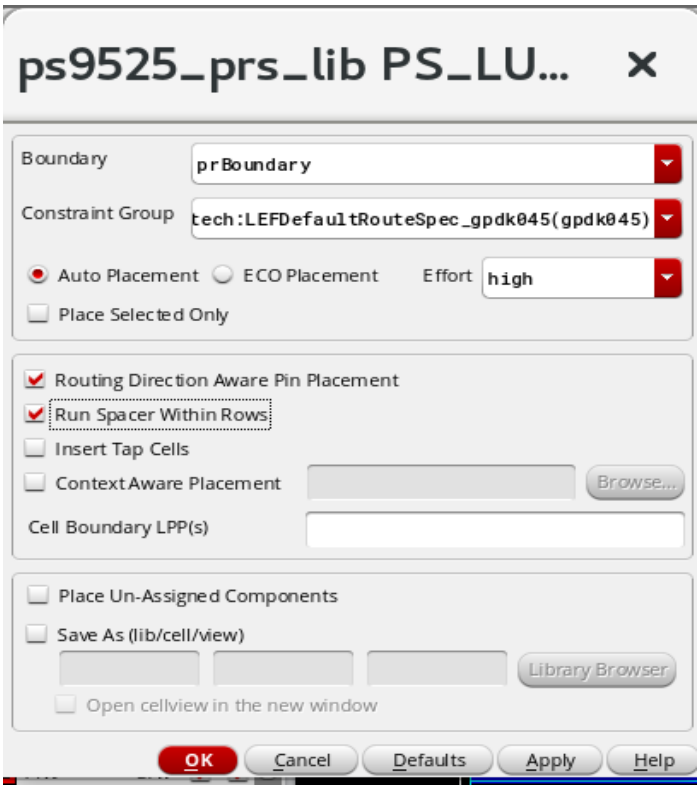

Figure 29. Auto Placement

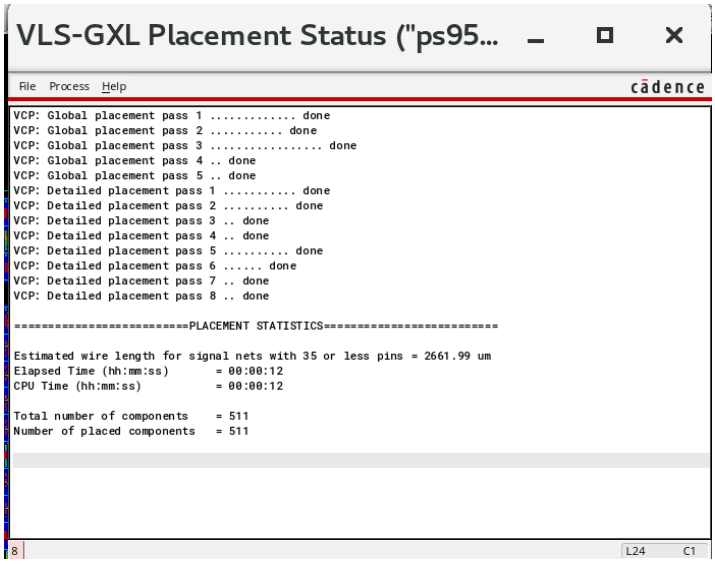

Figure 30. Auto Placement Results

After this step, the design will be placed within the PR region with 100% occupancy and the placement status will appear as shown in figure 30. All basic blocks are arranged as shown in figure 31.

Note: This layout shown over here is the final layout.

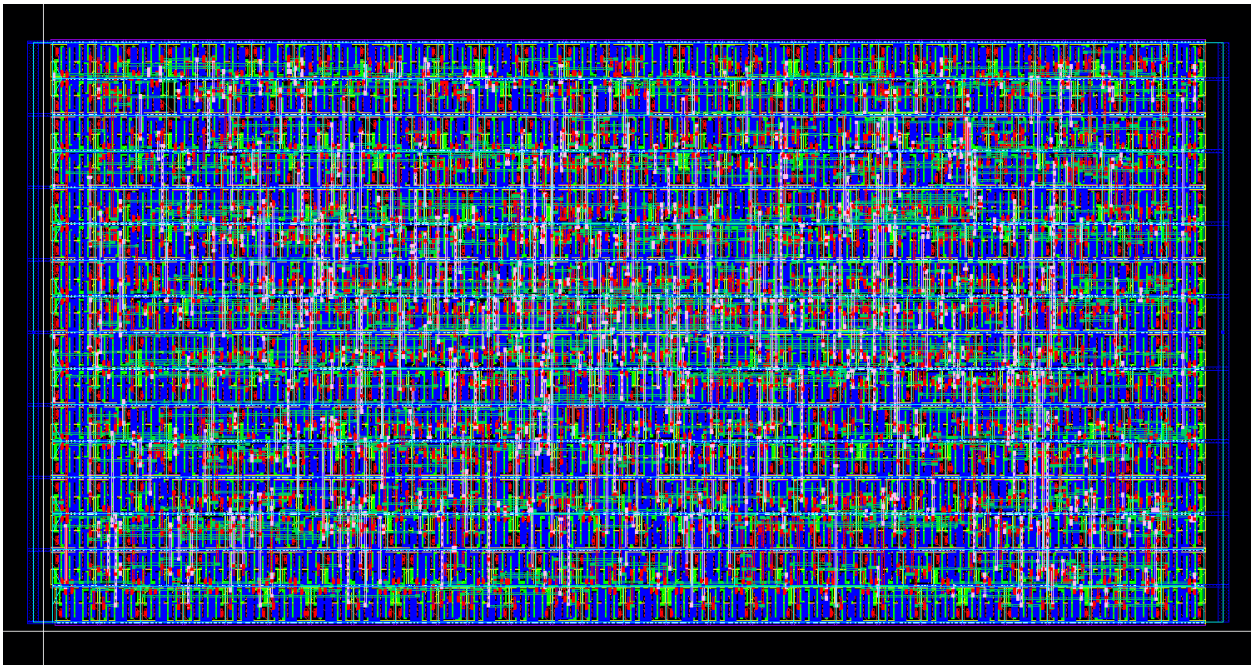

Figure 31. Final DUT layout

Before moving to the next steps, placement of power rails VDD and VSS on the sides will be a good idea. As can be seen in figure 32, power rails with 15 λ as the width generated at both sides and connected to each horizontal rail in the layout. Also, extending the PR region to the middle of the Power rails.

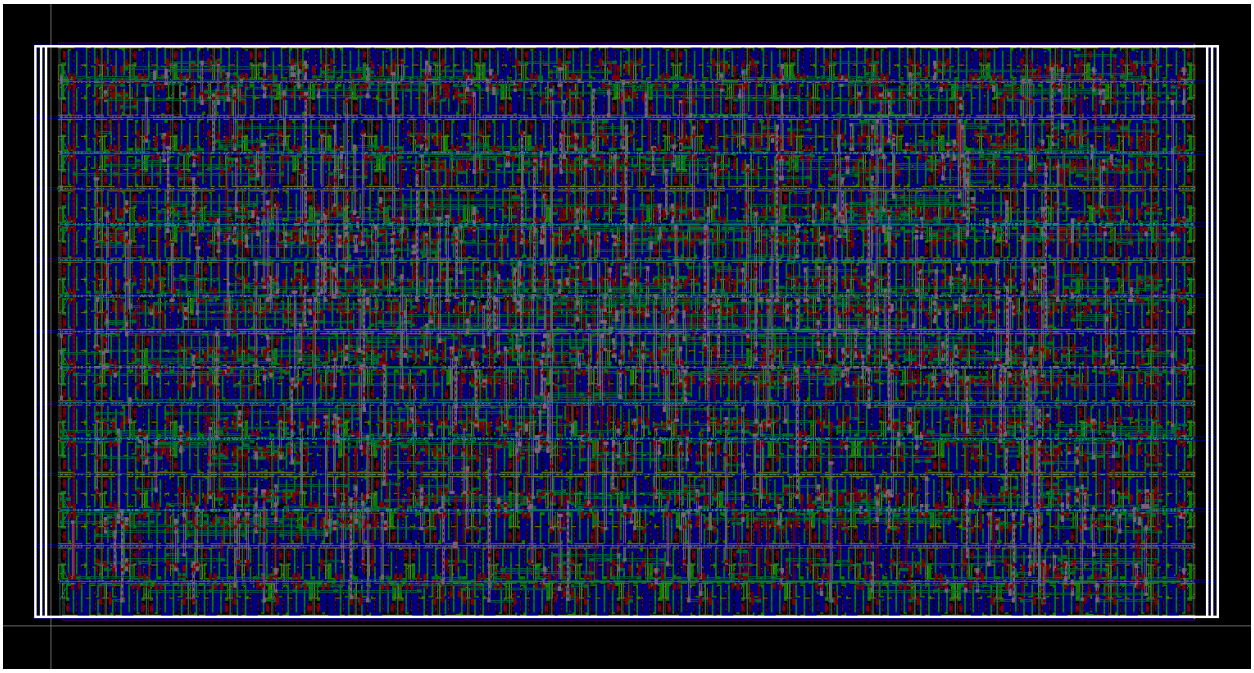

Figure 32. Final DUT layout with VDD and VSS highlighted

6.6 Pin arrangement

Generally, Pins are placed within the PR region. Open Place > Pin Placement. VDD and VSS pins are placed on the sides. Input pins are placed at the top, output pins are arranged at the bottom. Also, the placement depends on the layout requirement. For this layout, the area is a major concern. So, try to place all the input and output pins at the top of the pins label by using the level 1 pin option as shown in figure 33. After arranging all the pins are per the requirement, close the window.

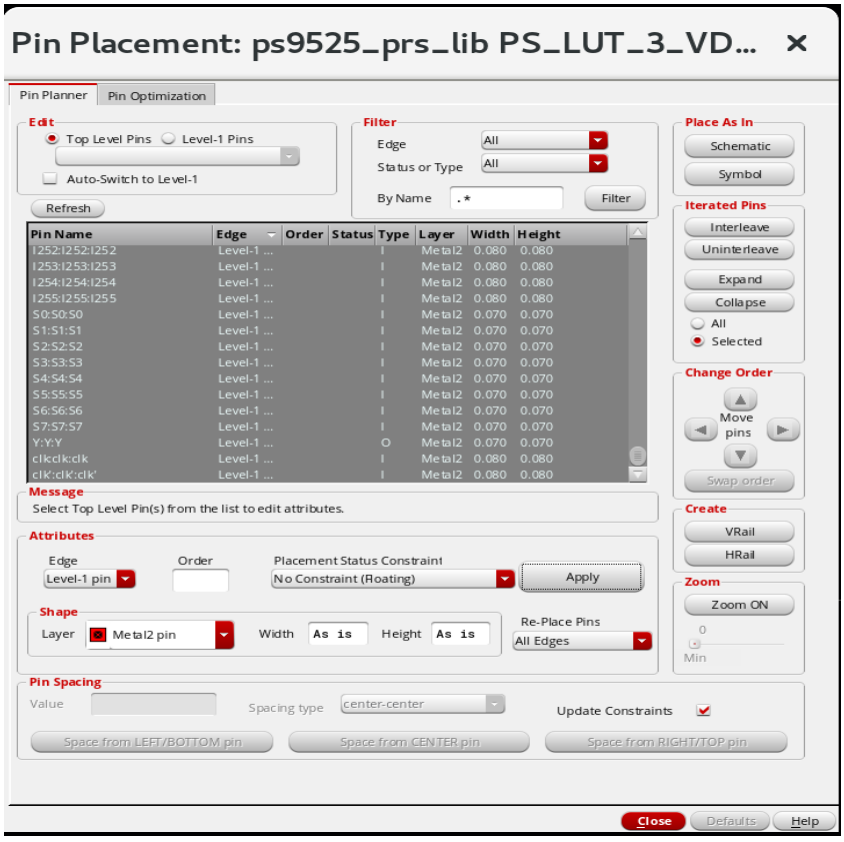

Figure 33. Pin Placement

6.7 Wire planning

Open Window > Assistant > Wire Assistant. A new wire assistant window will pop up on the right side. As shown in figure 34, start with the assigning base constraint group in the wire assistant. In this layout, only 4 metals will be utilized to do the layout. So make the changes as shown for the Top and Bottom metal layer.

The next step is the direction of the metals. Metal 1 and Metal 3 are assigned horizontal. Metal 2 and Metal 4 are assigned vertical direction. This will make the routing work easier and changes of shorting metals will also reduce. In this layout, only 4 metal layers are used for routing.

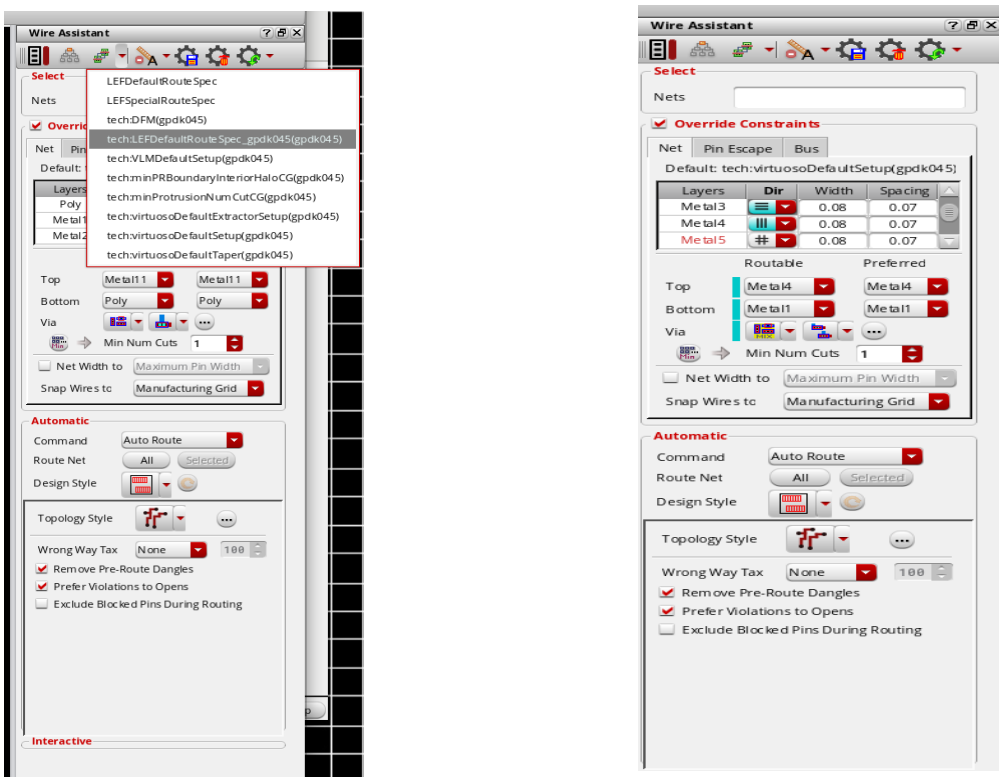

Figure 34. Wire Placement setting

For the via, there are three options to be filled. So choose Automatic Enclosure and Cut Direction & Offset or Center to the wire for the first two vias. For third via, make necessary changes as shown in figure 35.

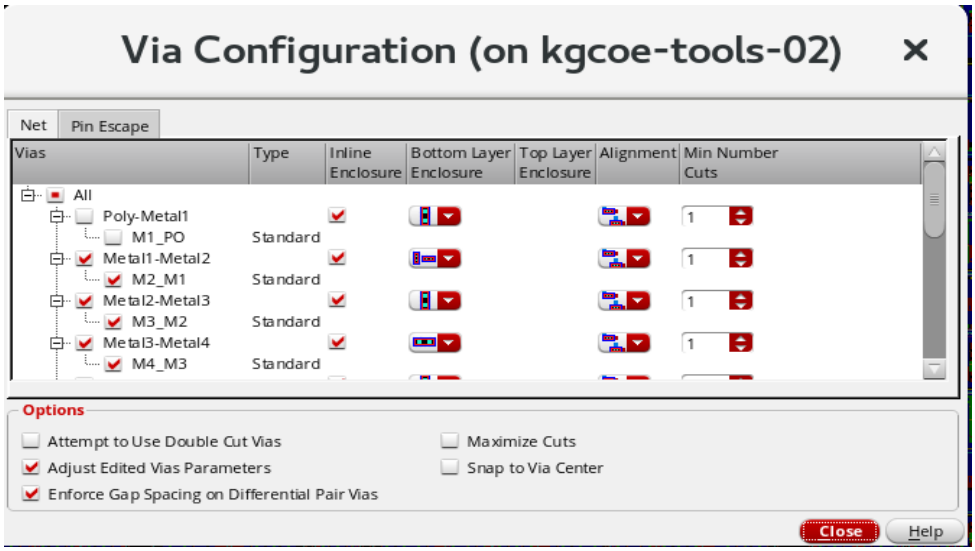

Figure 35. Via configuration - Wire Placement

Set the snap wire setting to Manufacturing Grid. The next step is to set the routing flow by changing the design style to ASIC. Routing technology can be selected by choosing the Minimum spanning Tree in the topology style options. Select the Wrong Way Tax button to none to restrict the routing from the opposite direction. Finally check the first two options named Remove Pre Route Dangles and Prefer violation to opens.

All setting for wire routing is done. To do automatic wire routing, click the All Route button. That will lead to the most optimum solution of routing. This process will take a significant amount of time depending on the complexity of the layout.

Check CIW window, if any shorts or open are observed in the final layout, it can be fixed by using Command Fix Violation and Command optimization in the Command option. Sometimes this command needs to run multiple times to reduce the errors.

Sometimes it is a good idea to check the routing for a single wire using the selected option in the Route net. To utilize this feature, select any pin or wire from the schematic/layout. Since a particular wire is chosen, the selected option is enabled and changes can be made for a particular wire. This feature is really useful when working in a complex environment.

The final step to finish the routing is getting the Design rule check (DRC) and Layout versus schematic (LVS) clean.

6.8 Setting for DRC and LVS

Open PVS > DRC run. As shown in figure 36, Set the path in the run Directory and other changes. Next select rules. Set the technology mapping file as

classes/ee620/tech/cds/gpdk045/GPDK045/TECH/GPDK045/gpdk045/pvtech.lib

Also, make necessary changes as shown in figures 36 & 37.

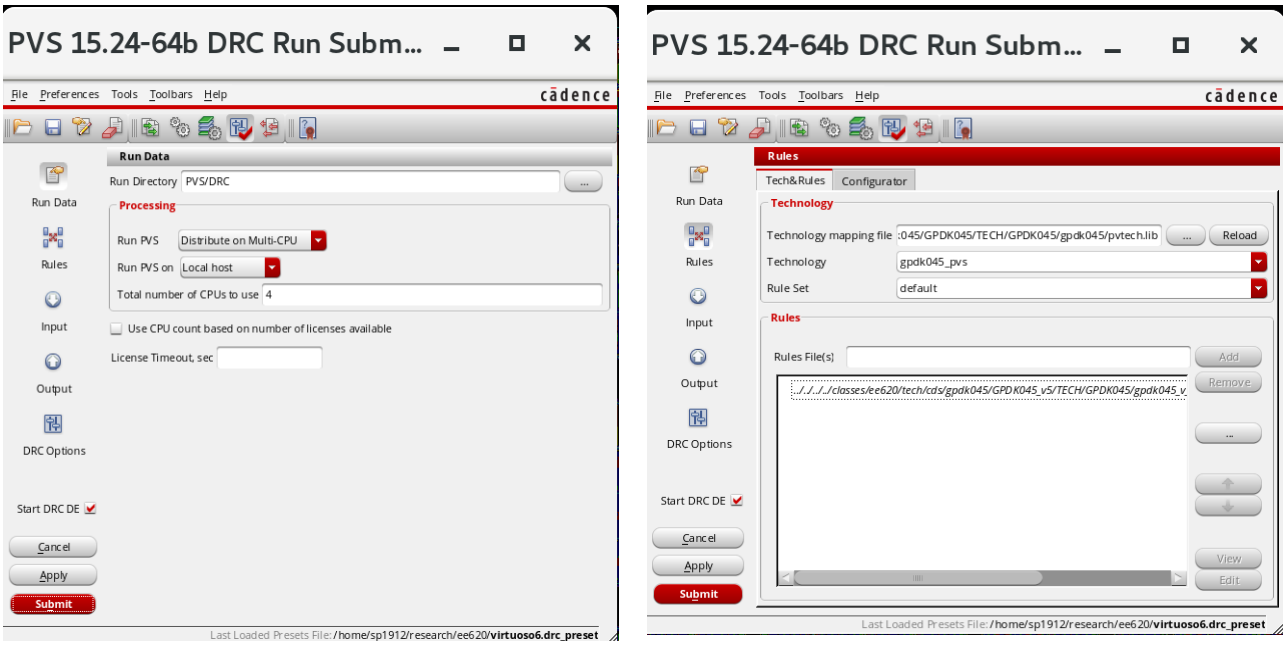

Figure 36. DRC setup 1 [5]

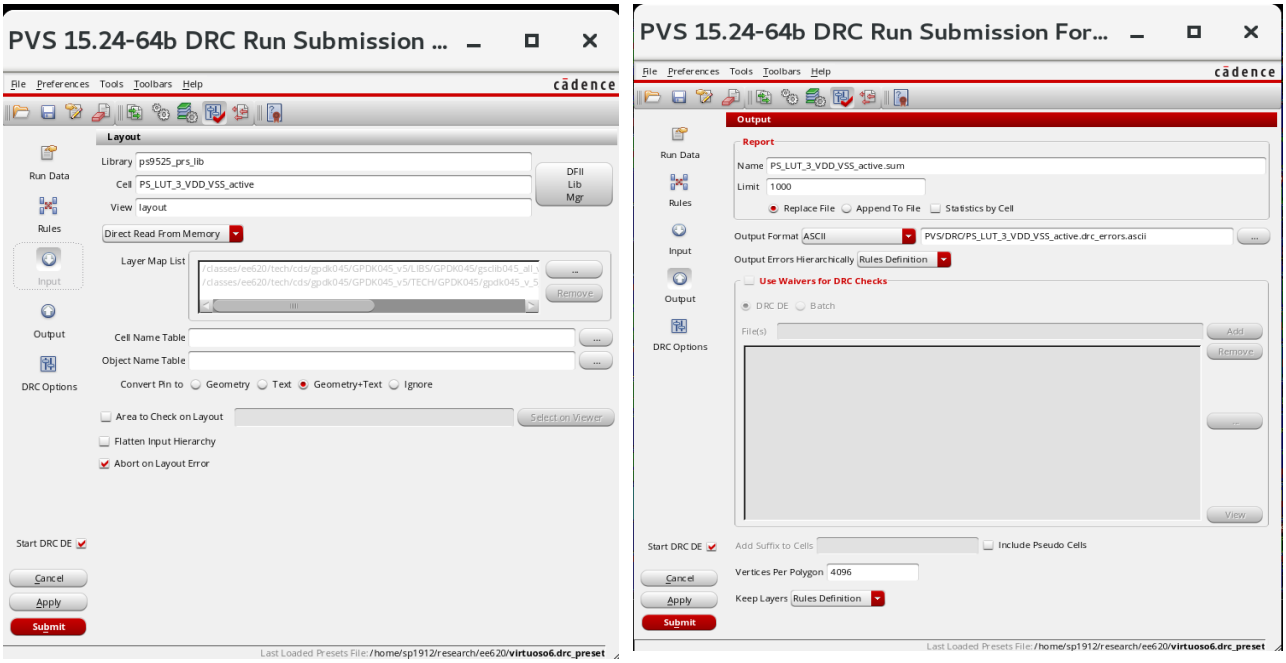

Figure 37. DRC setup 2[5]

No changes in the DRC options. After doing all this setting, click on Submit which will run, and check the DRC errors. The goal is to make the layout of DRC clean. After DRC clean, the report can be observed as shown in figure 38.

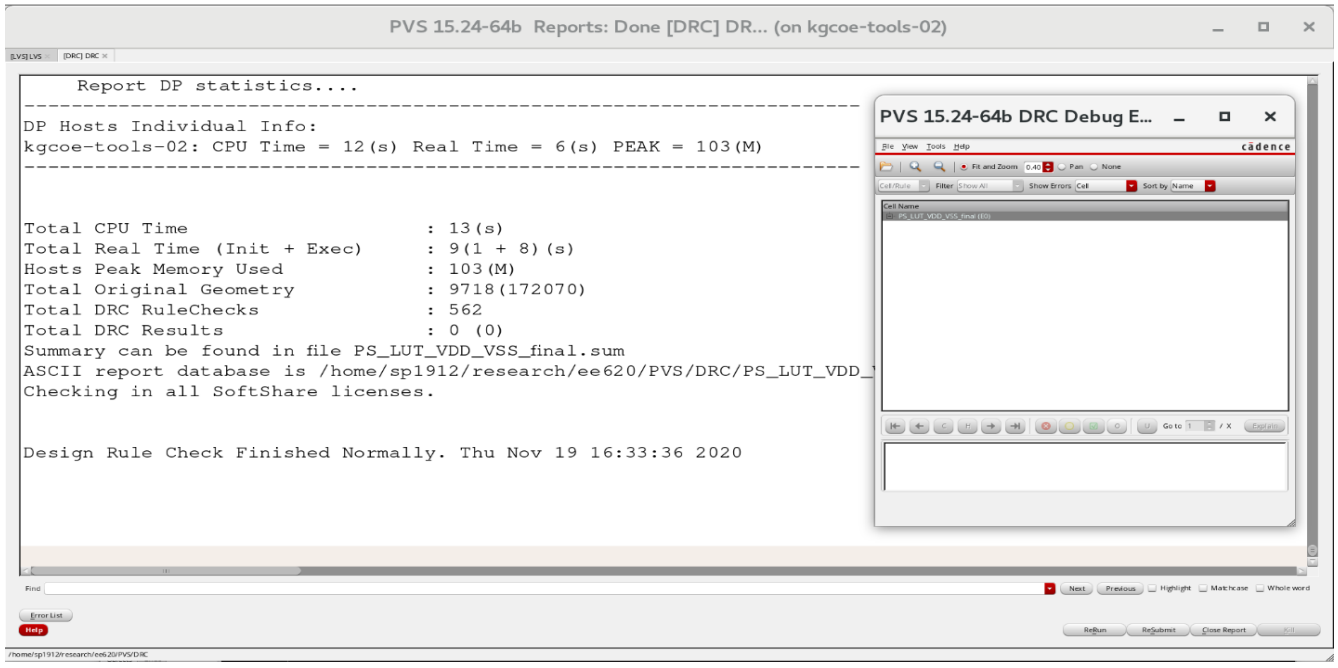

Figure 38. Final DRC Result

Similarly, the setting can be done for LVS analysis. Follow the next figure 39 to set the LVS and click Submit to run LVS.

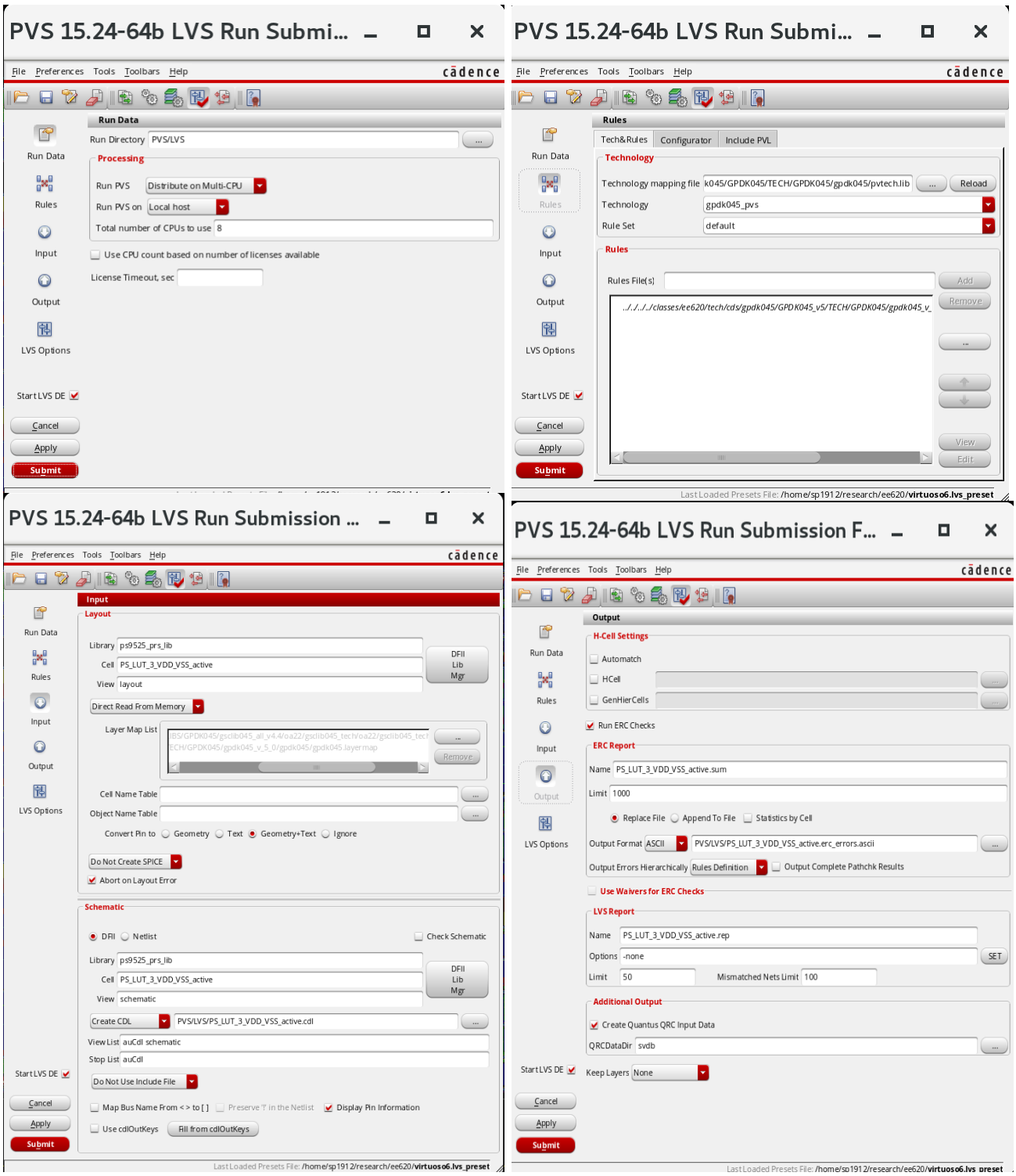

Figure 39. LVS setup [5]

LVS clean results are shown in figure 40.

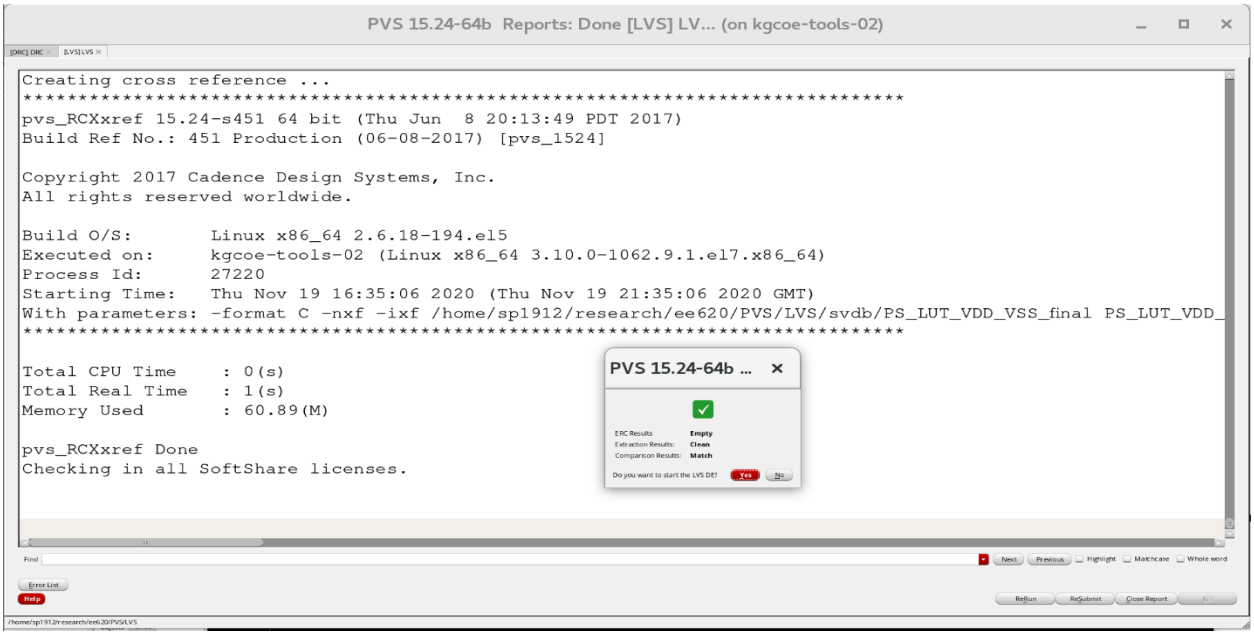

Figure 40. Final LVS result

The final layout of LUT is shown in figure 41.

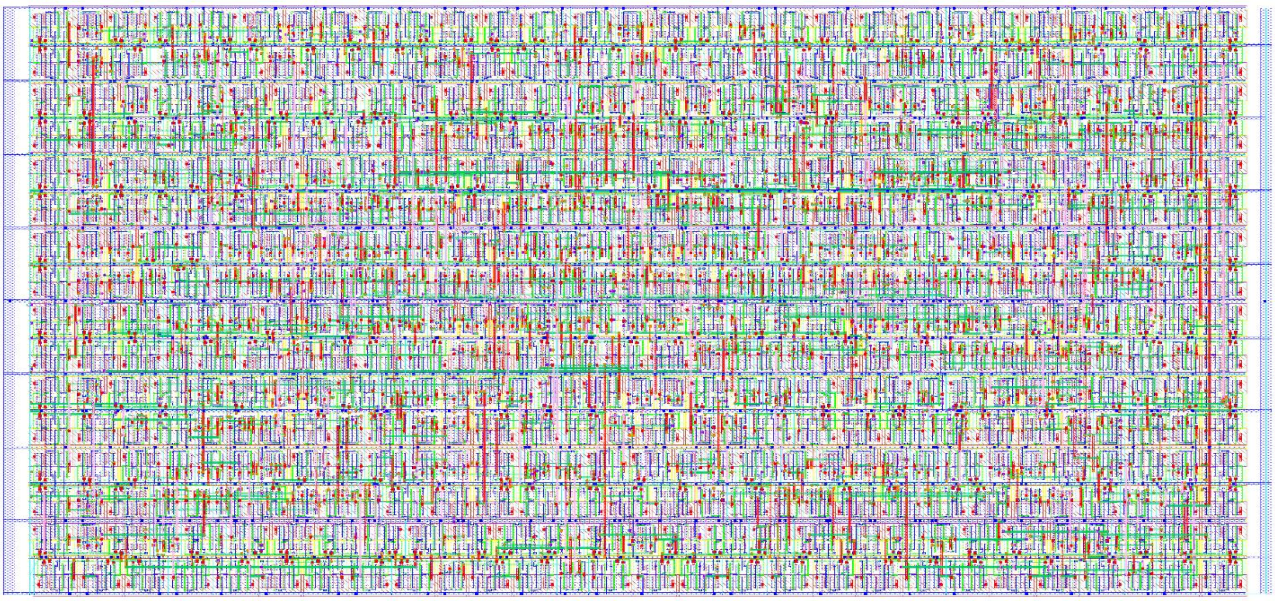

Figure 41. Final LUT layout

Area	54.46*27.6 μm^2

7. Core Design

Design of the Core layout is very tricky. To design a core, 8 LUTs, 8-bit register, and eight 8*1 MUX are required. So need to do the planning of the layout well before doing any simulation. At this stage, the layout of a 1 LUT is shown in figure 42.

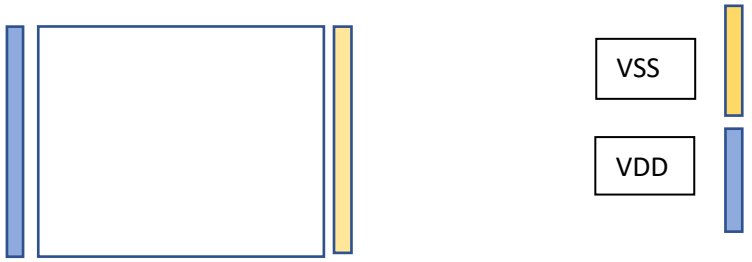

Figure 42. 1 LUT structure

Suppose similar kinds of 8 LUTs are used to make a core design, it will consume a more area by doubling the VDD and VSS rails as shown in figure 43.

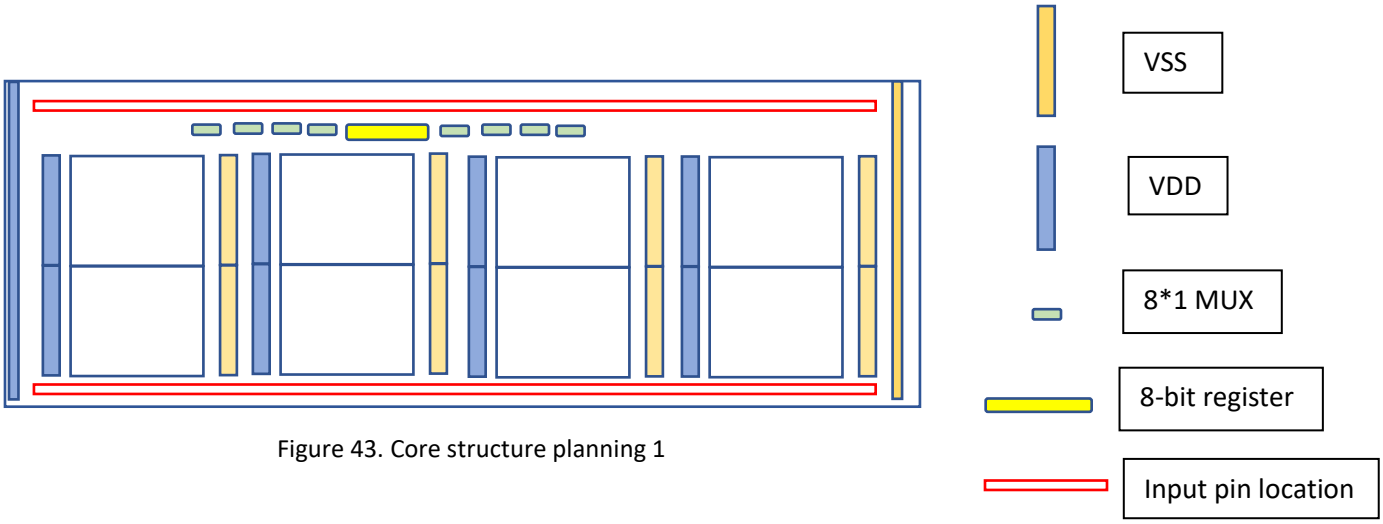

Figure 43. Core structure planning 1

Note: Dimensions are for rough ideas for all blocks shown in figures 43 and 44.

To reduce that area, consume, make another layout with just VDD on the right side and VSS on the left side and the final layout can be predicted as shown in figure 44.

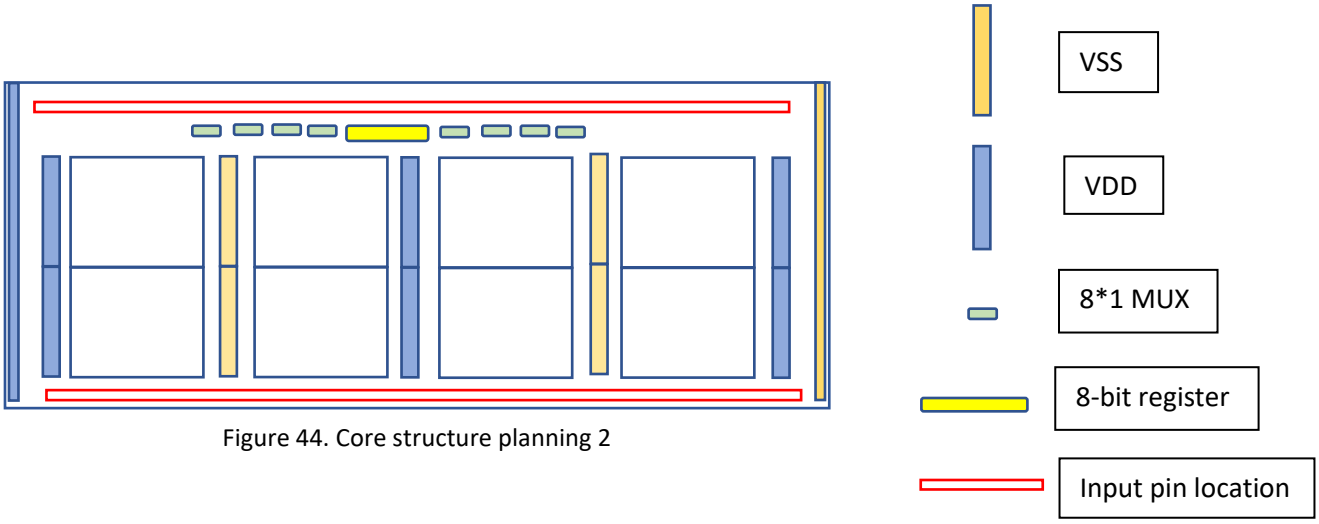

Figure 44. Core structure planning 2

7.1 Schematic of Core

Schematic of a core consist of 8 LUTs, eight 8*1 MUX, two 4-bit registers as shown in figure 45.

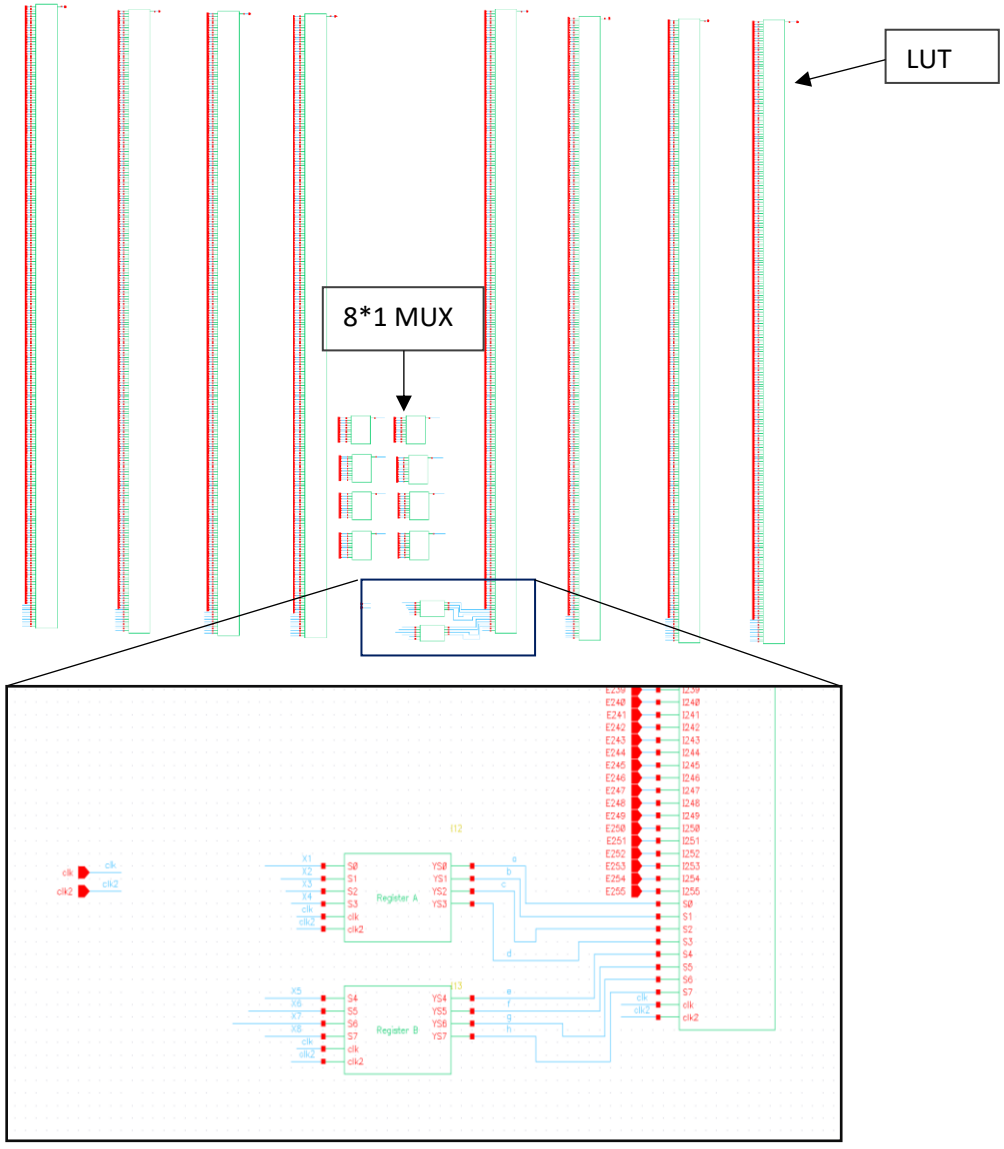

Figure 45. Core schematic

7.2 Core layout

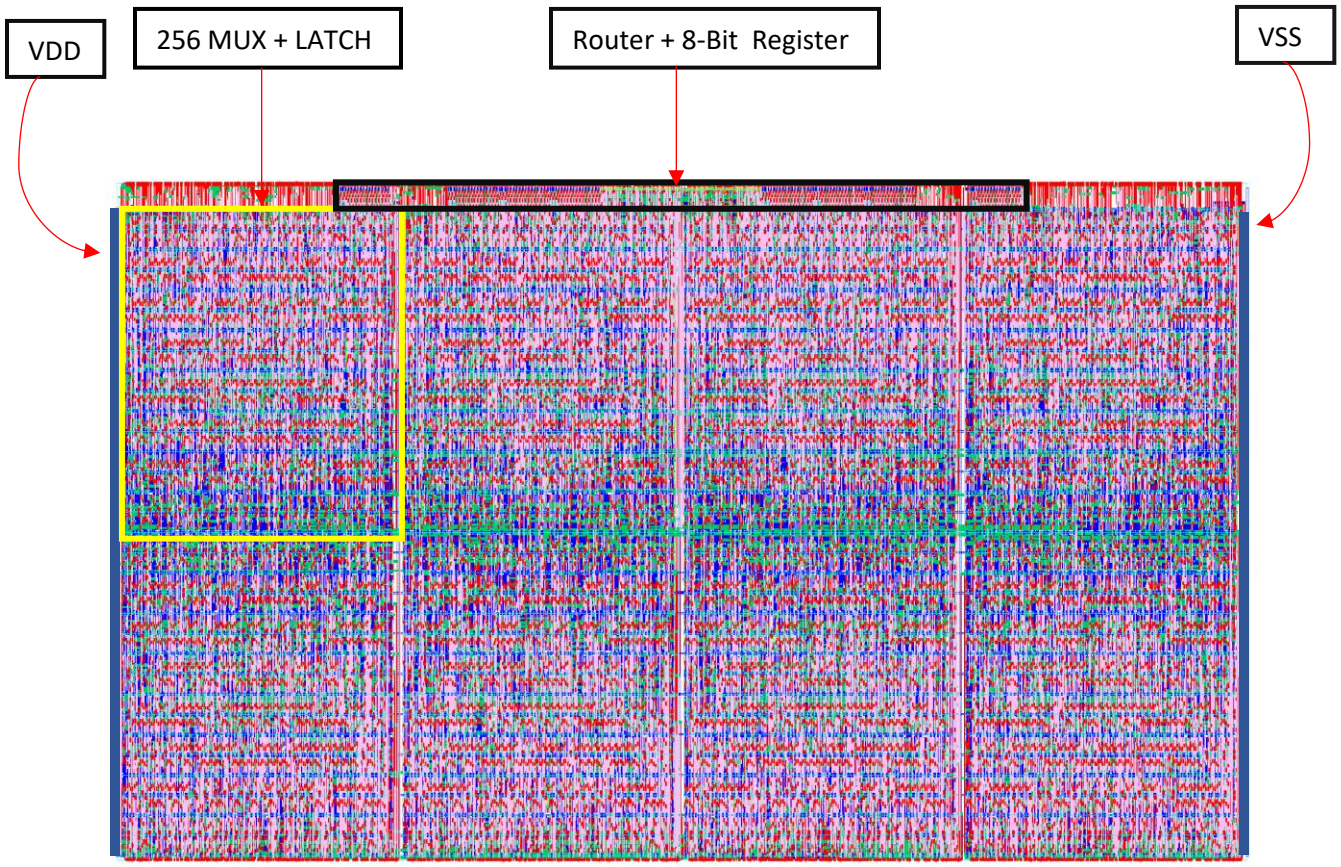

Figure 46. Core Layout

Area	226.6 * 57.4 μm^2

8. Cluster Design

Cluster design contains 9 core structure. By looking at the dimension of the subarray of DRAM, it will be a good idea to do the layout in 3*3 format. The layout of the cluster is shown in figure 47. But it still needed to be routed. At the current stage, the layout can give a general idea of the area.

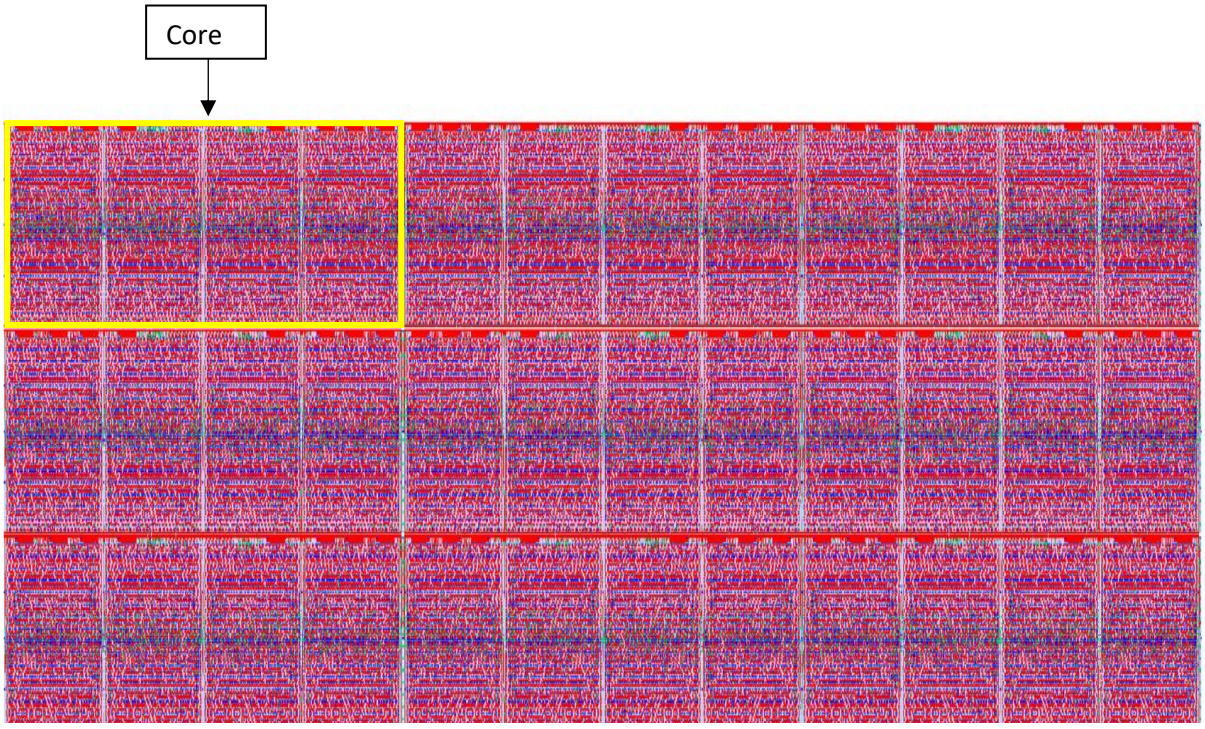

Figure 47. Cluster layout

Area	679.8 * 174.2 μm^2

9. Conclusion

This section consists of a brief report of work done and future work for the pPIM layout.

9.1 Observations

Common DRAM is chosen as a reference and started working on a possible solution for pPIM architecture to be fit. Starting from basic blocks like MUX, Latch, register, schematics are made followed by DRC and LVS clean layout.

From the DRAM core area, tried to make a very similar dimension layout for compatibility. With the core dimensions generated by pPIM, 8 clusters can be fit in a DRAM subarray. Cores of a cluster are arranged in a 3*3 matrix form in layout. All area dimensions are reported. The area of the core for pPIM is very near to the area calculated for 28nm DRAM.

9.2 Future work

Future work consists of routing for cluster design and measuring different delays using the AMS simulator. There is always room for reducing the layout using different methods so other work also aims to reduce the area of a core.

10. Reference

[1] P. Sutradhar, S. Bavikadi, M. Connolly, S. Prajapati, M. Indovina, S.Dinakarrao, A. Ganguly. Look-up-Table based processing-in-Memory Architecture with Programmable Precision-Scaling for deep learning Applications

[2] S. Li, D. Niu, K. Malladi, H.Zheng, B.Brennan, Y.Xie. DRISA: A DRAM-based Reconfigurable In-Situ Accelerator

[3] Cadence Reference Manual Generic 45nm Silicide 1.0V/1.8V 1P 11M Process Design Kit and Rules decks (PRD) Revision 3.5

[4] M. Indovina. EDA Tutorial 2 on AutoRoute (EE620-RIT)

[5] M. Indovina EDA Tutorial 1 Basic Design and layout information in Cadence Virtuoso (EE620-RIT)