Project Report on RTL to GDS: ASIC Design Flow

Implementation of K-Means Clustering Circuit using RTL to GDSII Flow

Group Members: Rahul Rathore (2024PVL0100),

Guide: Dr. Rohit Chaurasiya
Institute: Indian Institute of Technology Jammu
Department: Electrical Engineering
Lab: IC-ResQ Lab

Abstract

K-Means clustering is a widely used machine learning technique that groups similar data points into clusters. In the context of the ASIC design flow, this technique can significantly improve stages such as logic synthesis and physical design. Tools like Genus (for synthesis) and Innovus (for placement and routing) benefit from clustering by organizing related logic blocks closer together. This optimization reduces wire length, enhances timing performance, lowers power consumption, and accelerates the overall design process. By minimizing design complexity and improving EDA tool efficiency, K-Means clustering proves to be a valuable asset in modern VLSI design.

1 Tools Used

- Vivado (for RTL simulation and schematic)
- Cadence Genus (for logic synthesis)
- Cadence Innovus (for placement, CTS, routing)
- NCLaunch (for simulation)

2 Introduction

ASIC design flow includes several important steps, such as synthesis and physical design. Genus is a tool used for synthesis, where RTL code is converted into a gate-level netlist.

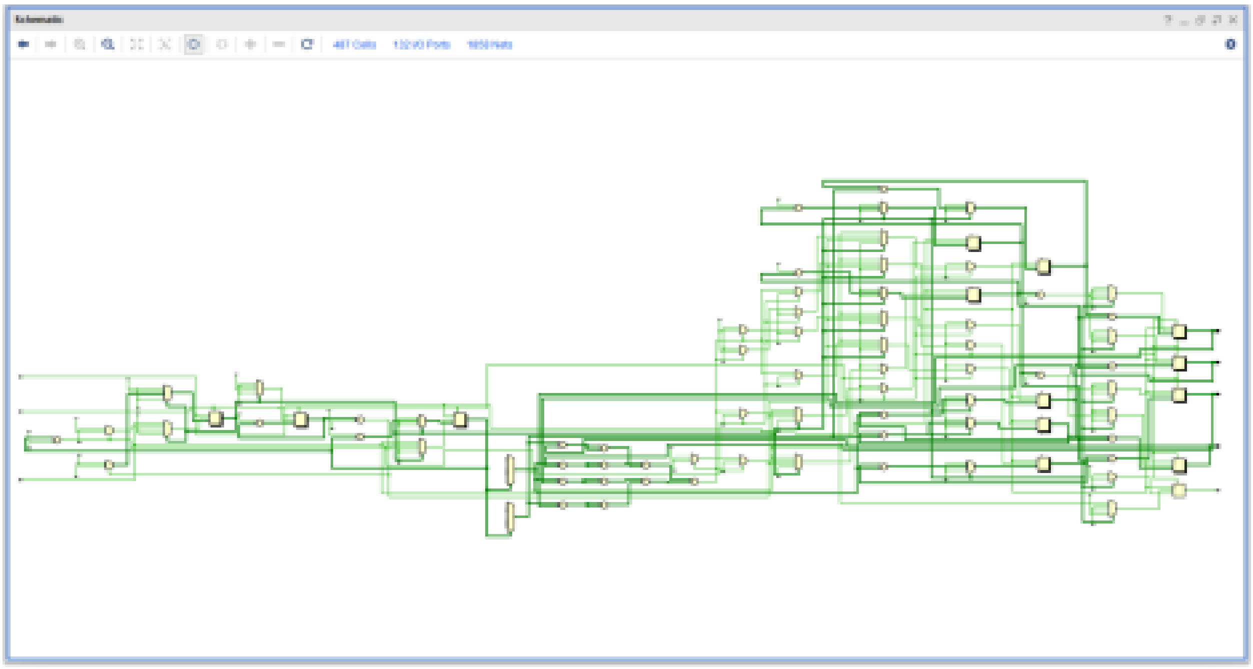


Figure 1: Vivado Elaborated Design

Scripting in Genus is done using TCL (.tcl) files to automate the flow. A slow library file is used to ensure the design meets timing in the worst-case scenario. After synthesis, the design moves to physical design using tools like Innovus. This stage includes placement and routing. Optimizing this flow using techniques like K-Means clustering can help improve performance, reduce power, and manage complexity in large chip designs.

3 RTL Designing And Simulation Testing:

Step 1 : Open the Vivado tool, save the code in the .v format in clustering which is Machine Learning Algorithm and Get the schematic.
Step 2 : Write the testbench of the following code in .v format check the testing of the code in the simulation Behavior Tool of Vivado.
Note: Here we are trying to get check the RTL design in which we get the netlist according to the tool Optimization technique where we get the schematic and use the logical tool and with different simulation test vectors we can check the functionality of the tool.

4 Genus Tool Usage:

Genus is a tool from Cadence used for logic synthesis in ASIC design. Genus takes your RTL code (like Verilog or VHDL) and converts it into a gate-level netlist using standard cells. It checks timing, area, and power, and lets you write constraints using .sdc files or automate tasks with .tcl scripts. Basically, it's the first step in turning your code into real hardware for physical design tools like Innovus.

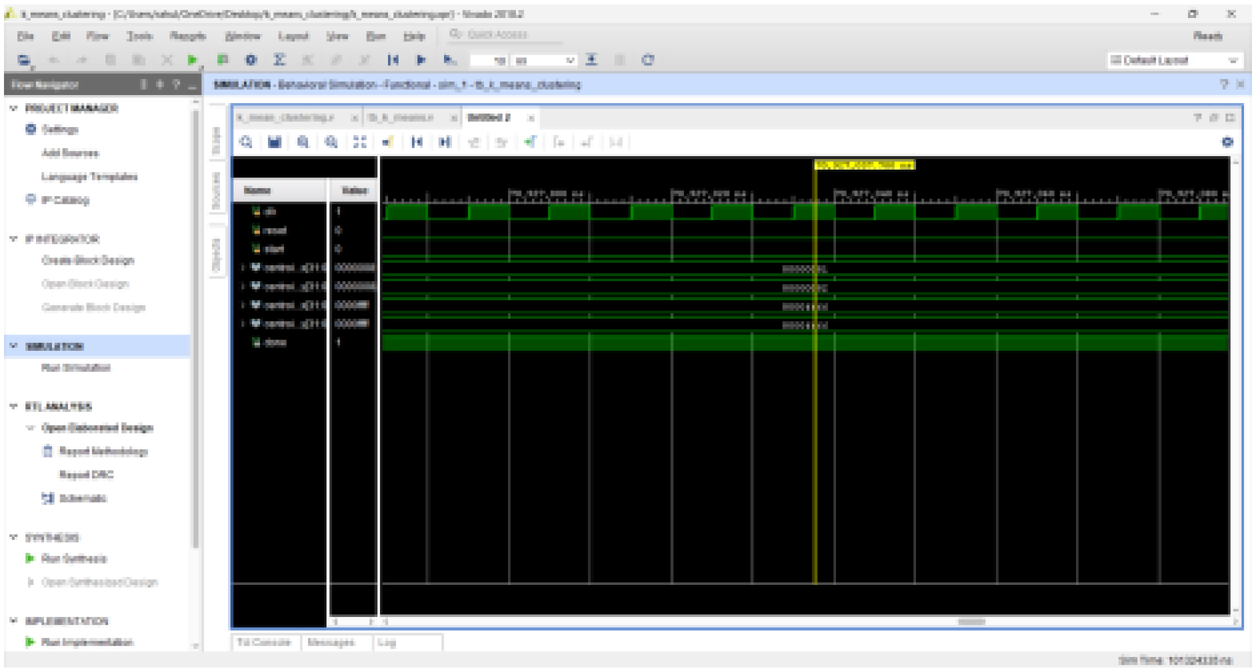


Figure 2: Vivado Simulation Test

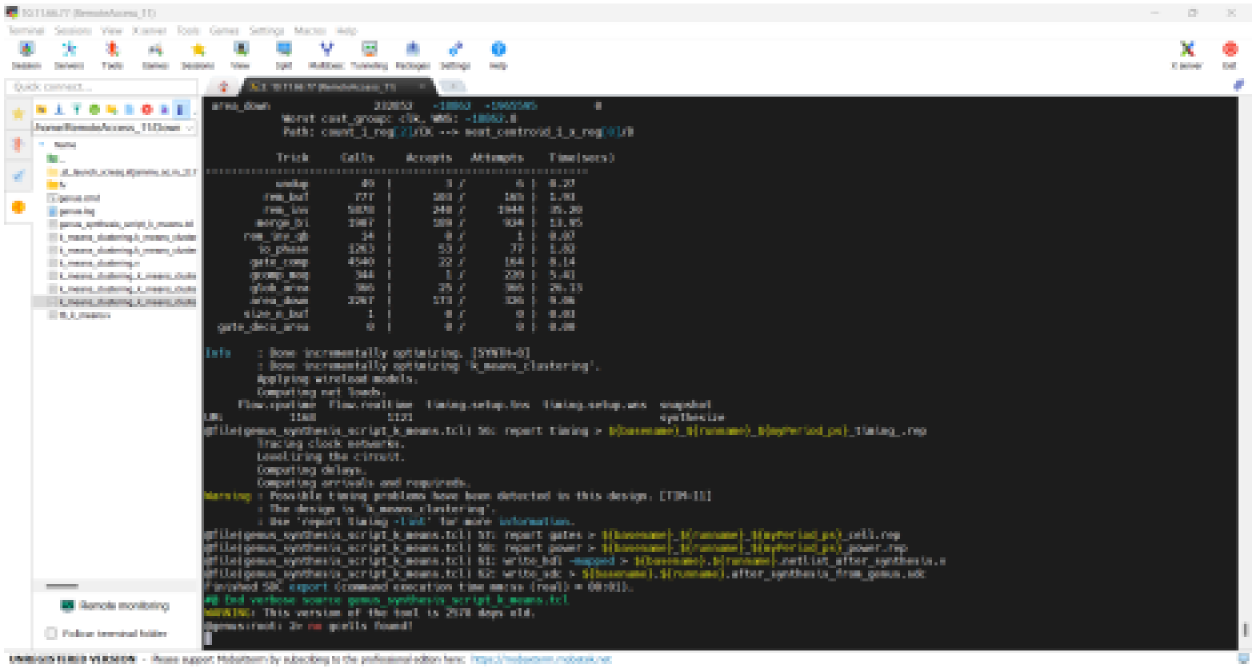
We use Genus Tool to get the synthesis netlists in .v format according to the library files which have been provided by the foundry in which we generally use slow.lib or fast.lib.

TCL file is a script used to automate commands in tools like Genus or Vivado. Instead of clicking buttons, you write steps like reading files, setting clocks, and running synthesis. It saves time and makes the design flow repeatable and easy to debug.

After the successful running of tcl it generates the following report which shows the functionality and brief of the circuit constraints design as well as the reports of the provided information which are important for the designing.
In ASIC design, .lib files are timing library files provided by the foundry (e.g., TSMC, GlobalFoundries, etc.). They describe the timing, power, and functional behavior of stan dard cells at different conditions. These files are used during synthesis, timing analysis, and place-and-route.

slow.lib (also called worst-case corner): This file models the slowest performance of the standard cells — for example, when voltage is low and temperature is high. It ensures the chip works reliably in the worst-case conditions (slow speed, highest delays). Used for setup timing checks.

Fast.lib (also called best-case corner): This file models the fastest performance — for example, when voltage is high and temperature is low. It's used to check for hold violations, where signals arrive too early. Used for hold timing checks.



```
area_down              232852   -18862   -1965585        0
          Worst cost_group: clk, WNS: -18862.8
          Path: count_1_reg[2]/CK --> next_centroid_1_x_reg[0]/D

          Trick      Calls     Accepts   Attempts     Time(secs)
-----------------------------------------------------------------
          undup        49  |        3 /        6 |   0.22
        rem_buf       777  |      383 /      165 |   1.93
        rem_inv      5878  |      248 /     1944 |  35.38
       merge_bi      1987  |      389 /      934 |  13.95
     rem_inv_qb        34  |        0 /        1 |   0.02
       io_phase      1263  |       53 /       77 |   1.02
      gate_comp      4540  |       22 /      184 |   8.14
      gcomp_mog       344  |        1 /      220 |   5.41
      glob_area       386  |       25 /      386 |  26.13
      area_down      2267  |      373 /      326 |   9.06
     size_n_buf         1  |        0 /        0 |   0.03
 gate_deco_area         0  |        0 /        0 |   0.00

Info    : Done incrementally optimizing. [SYNTH-8]
        : Done incrementally optimizing 'k_means_clustering'.
        Applying wireload models.
        Computing net loads.
      Flow.cputime  Flow.realtime  timing.setup.tns  timing.setup.wns  snapshot
UM:          1168           1121                                       synthesize
@file(genus_synthesis_script_k_means.tcl) 56: report timing > ${basename}_${runname}_${myPeriod_ps}_timing_.rep
        Tracing clock networks.
        Levelizing the circuit.
        Computing delays.
        Computing arrivals and requireds.
Warning : Possible timing problems have been detected in this design. [TIM-11]
        : The design is 'k_means_clustering'.
        : Use 'report timing -lint' for more information.
@file(genus_synthesis_script_k_means.tcl) 57: report gates > ${basename}_${runname}_${myPeriod_ps}_cell.rep
@file(genus_synthesis_script_k_means.tcl) 58: report power > ${basename}_${runname}_${myPeriod_ps}_power.rep
@file(genus_synthesis_script_k_means.tcl) 61: write_hdl -mapped > ${basename}.${runname}.netlist_after_synthesis.v
@file(genus_synthesis_script_k_means.tcl) 62: write_sdc > ${basename}.${runname}.after_synthesis_from_genus.sdc
Finished SDC export (command execution time mm:ss (real) = 00:01).
#@ End verbose source genus_synthesis_script_k_means.tcl
WARNING: This version of the tool is 2578 days old.
@genus:root: 2> no gcells found!
```

Figure 3: Reports Generated After Genus

.v (Netlist File): The .v file is a Verilog netlist generated after synthesis, containing gate level connections of the design. It shows how standard cells are connected but no longer includes high-level logic like if or case statements

.sdc (Synopsys Design Constraints): The .sdc file defines design constraints like clock definitions, input/output delays, and timing exceptions. It guides the synthesis and place and-route tools to meet timing and performance goals.

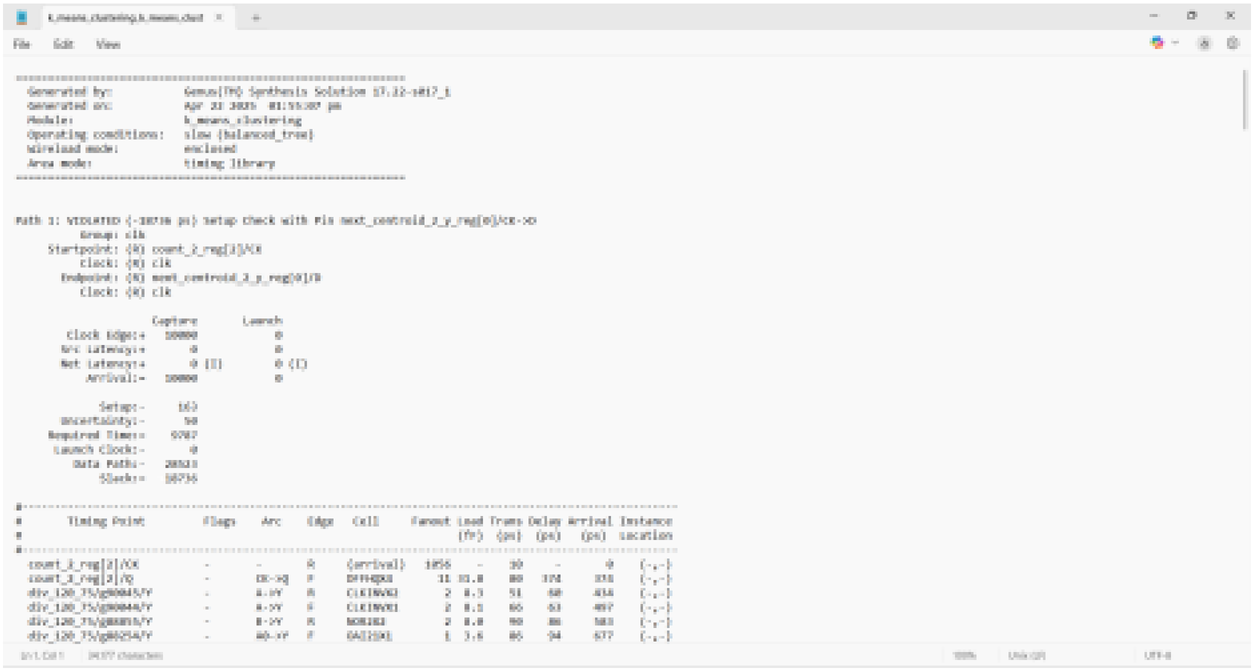
Timing report checks if the design meets the required timing constraints, such as a 10 ps or 50 ps clock period. It shows setup and hold timing for all paths, helping to ensure the circuit works correctly at the target speed.

Cell report lists all the standard cells used after synthesis and gives details like cell delay, area, and function. With tighter timing (e.g., 10 ps), faster and sometimes larger cells may be used to meet timing.

Power report shows how much power the chip consumes, including dynamic, static, and leakage power. Designs targeting faster speeds like 10 ps usually consume more power due to increased switching and use of higher drive-strength cells.

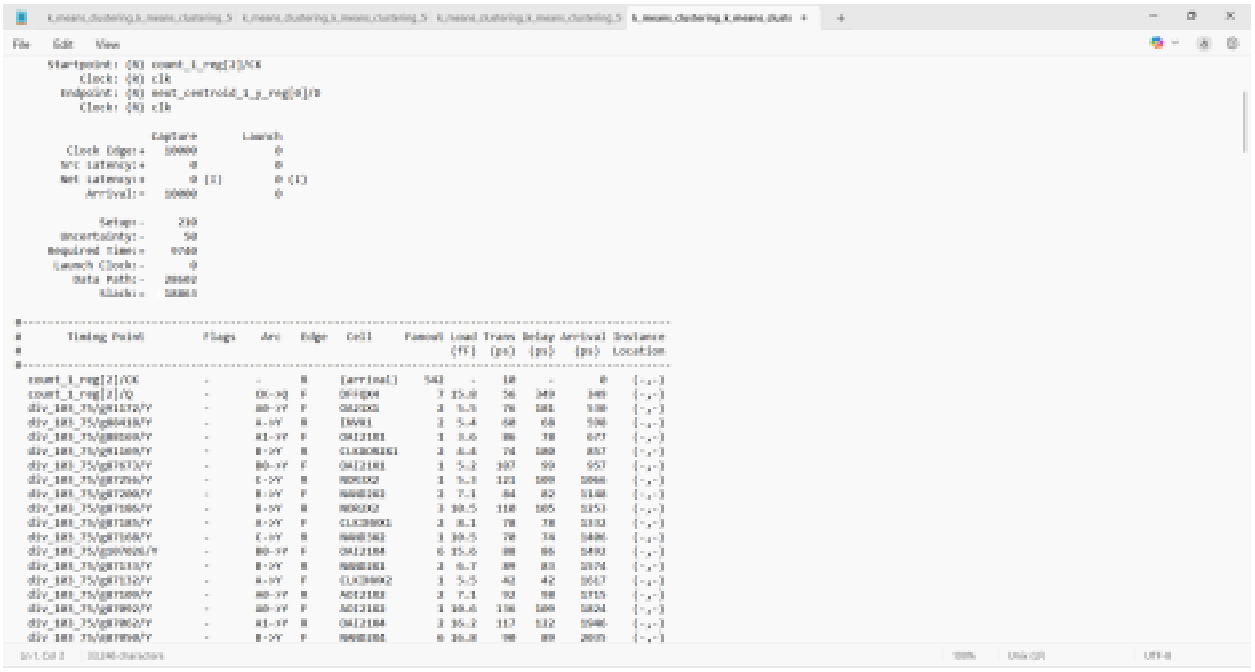
5 NCLAUNCH

nclaunch is a graphical user interface (GUI) tool in Cadence used to manage simulation setups for digital designs.



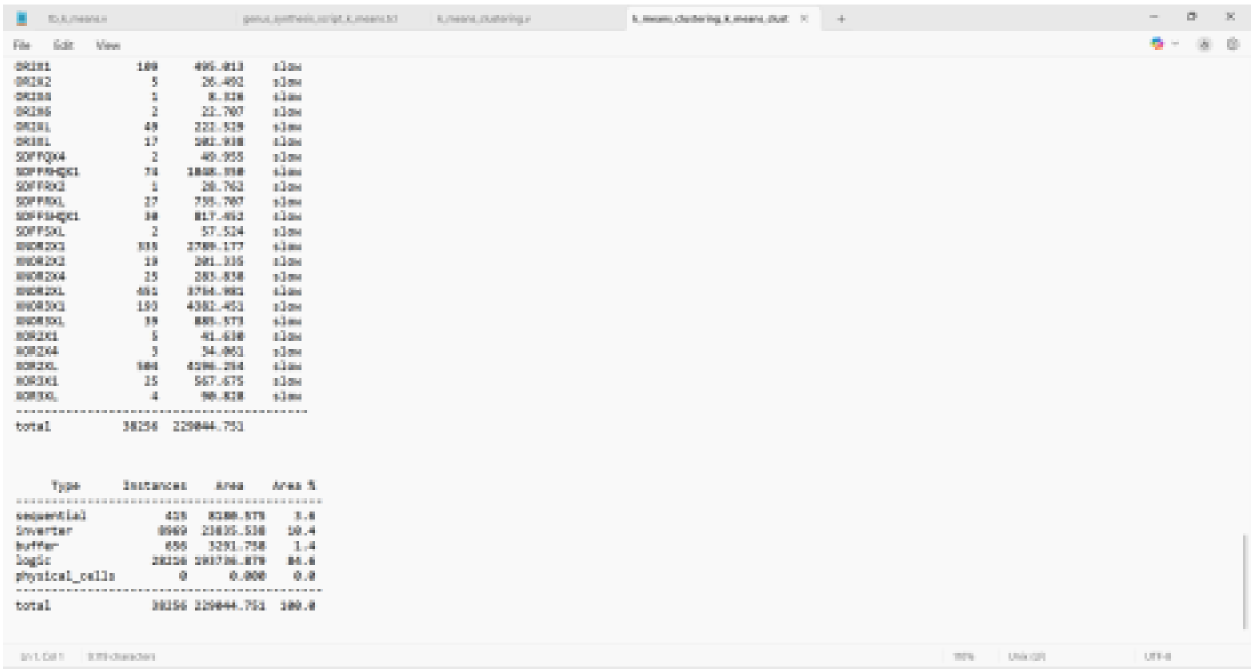
```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Apr 22 2025  01:55:07 pm
  Module:                 k_means_clustering
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================


Path 1: VIOLATED (-10736 ps) Setup Check with Pin next_centroid_2_y_reg[0]/CK->D
          Group: clk
     Startpoint: (R) count_2_reg[2]/CK
          Clock: (R) clk
       Endpoint: (R) next_centroid_2_y_reg[0]/D
          Clock: (R) clk

                     Capture       Launch
       Clock Edge:+    10000            0
      Src Latency:+        0            0
      Net Latency:+        0 (I)        0 (I)
          Arrival:=    10000            0

            Setup:-      163
      Uncertainty:-       50
    Required Time:=     9787
     Launch Clock:-        0
        Data Path:-    20523
            Slack:=   -10736

#---------------------------------------------------------------------------------------------------------
#      Timing Point         Flags     Arc     Edge    Cell         Fanout  Load  Trans  Delay  Arrival  Instance
#                                                                          (fF)   (ps)   (ps)    (ps)   Location
#---------------------------------------------------------------------------------------------------------
  count_2_reg[2]/CK          -         -       R     (arrival)     1056     -     30      -        0    (-,-)
  count_2_reg[2]/Q           -        CK->Q    F     DFFHQX1         11  11.8     80    374      374    (-,-)
  div_120_75/g00045/Y        -        A->Y     R     CLKINVX2         2   8.3     51     60      434    (-,-)
  div_120_75/g00044/Y        -        A->Y     F     CLKINVX1         2   8.1     66     63      497    (-,-)
  div_120_75/g00015/Y        -        B->Y     R     NOR2X1           2   8.0     90     86      583    (-,-)
  div_120_75/g00254/Y        -        A0->Y    F     OAI21X1          1   3.6     85     94      677    (-,-)
```

Figure 4: Timing Report @10



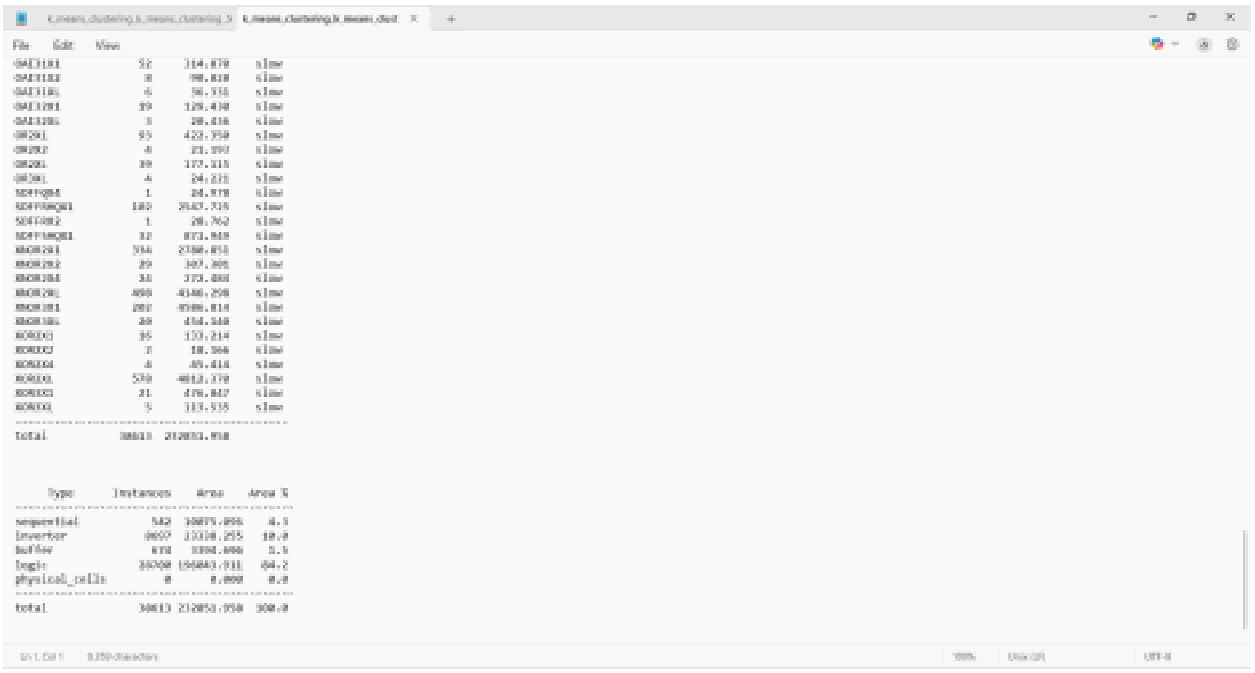
```
     Startpoint: (R) count_1_reg[2]/CK
          Clock: (R) clk
       Endpoint: (R) next_centroid_1_y_reg[0]/D
          Clock: (R) clk

                     Capture       Launch
       Clock Edge:+    10000            0
      Src Latency:+        0            0
      Net Latency:+        0 (I)        0 (I)
          Arrival:=    10000            0

            Setup:-      210
      Uncertainty:-       50
    Required Time:=     9740
     Launch Clock:-        0
        Data Path:-    20603
            Slack:=   -10863

#-------------------------------------------------------------------------------------------------------
#      Timing Point         Flags     Arc    Edge    Cell         Fanout  Load  Trans  Delay  Arrival  Instance
#                                                                         (fF)   (ps)   (ps)    (ps)   Location
#-------------------------------------------------------------------------------------------------------
  count_1_reg[2]/CK          -         -      R     (arrival)      542     -     10      -        0    (-,-)
  count_1_reg[2]/Q           -        CK->Q   F     DFFQX4           7  15.0     56    349      349    (-,-)
  div_103_75/g01172/Y        -        B0->Y   F     OA21X1           2   5.5     76    181      530    (-,-)
  div_103_75/g00418/Y        -        A->Y    R     INVX1            2   5.4     60     68      598    (-,-)
  div_103_75/g00169/Y        -        A1->Y   F     OAI21X1          1   3.6     86     79      677    (-,-)
  div_103_75/g01169/Y        -        B->Y    R     CLKXOR2X1        2   4.4     74    180      857    (-,-)
  div_103_75/g07673/Y        -        B0->Y   F     OAI21X1          1   5.2    107     99      957    (-,-)
  div_103_75/g07256/Y        -        C->Y    R     NOR3X2           1   5.3    121    109     1066    (-,-)
  div_103_75/g07208/Y        -        B->Y    F     NAND2X2          2   7.1     84     82     1148    (-,-)
  div_103_75/g07186/Y        -        B->Y    R     NOR2X2           3  10.5    118    105     1253    (-,-)
  div_103_75/g07185/Y        -        A->Y    F     CLKINVX1         2   8.1     78     78     1332    (-,-)
  div_103_75/g07168/Y        -        C->Y    R     NAND3X2          1  10.5     78     74     1406    (-,-)
  div_103_75/g097026/Y       -        B0->Y   F     OAI21X4          6  15.6     88     86     1492    (-,-)
  div_103_75/g07133/Y        -        B->Y    R     NAND2X1          2   6.7     89     83     1574    (-,-)
  div_103_75/g07132/Y        -        A->Y    F     CLKINVX2         1   5.5     42     42     1617    (-,-)
  div_103_75/g07100/Y        -        A0->Y   R     AOI21X2          2   7.1     92     98     1715    (-,-)
  div_103_75/g07092/Y        -        B0->Y   F     AOI21X2          1  10.6    136    109     1824    (-,-)
  div_103_75/g07062/Y        -        A1->Y   R     OAI21X4          2  16.2    117    122     1946    (-,-)
  div_103_75/g07058/Y        -        B->Y    F     NAND2X4          6  16.8     90     89     2035    (-,-)
```

Figure 5: Timing Report @50



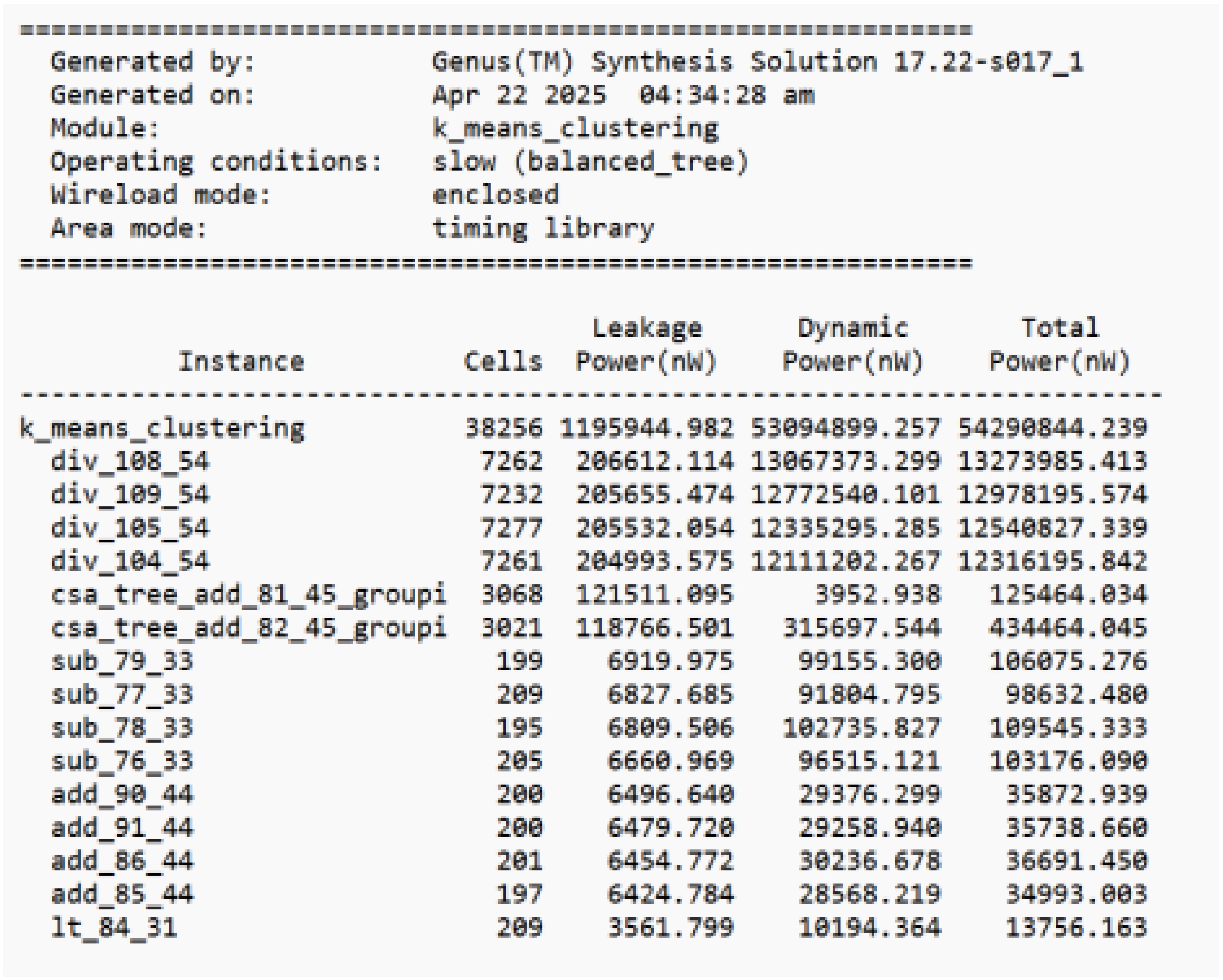
```
OR2X1            100     495.013    slow
OR2X2              5      26.492    slow
OR2X4              1       8.326    slow
OR2X6              2      22.707    slow
OR2XL             49     221.529    slow
OR3XL             17     102.938    slow
SDFFQX4            2      49.955    slow
SDFFRHQX1         74    1848.950    slow
SDFFRX2            1      28.762    slow
SDFFRXL           27     735.707    slow
SDFFSHQX1         30     817.452    slow
SDFFSXL            2      57.524    slow
XNOR2X1          335    2789.177    slow
XNOR2X2           19     201.335    slow
XNOR2X4           25     283.838    slow
XNOR2XL          451    3754.981    slow
XNOR3X1          193    4382.451    slow
XNOR3XL           39     885.573    slow
XOR2X1             5      41.630    slow
XOR2X4             3      54.063    slow
XOR2XL           504    4196.254    slow
XOR3X1            25     567.675    slow
XOR3XL             4      90.828    slow
-------------------------------------------
total          38256  229044.751


     Type      Instances     Area     Area %
---------------------------------------------
sequential           425   8180.576     3.6
inverter            8949  23835.538    10.4
buffer               656   3291.758     1.4
logic              28226 193736.879    84.6
physical_cells         0      0.000     0.0
---------------------------------------------
total              38256 229044.751   100.0
```

Figure 6: Cell Report @10



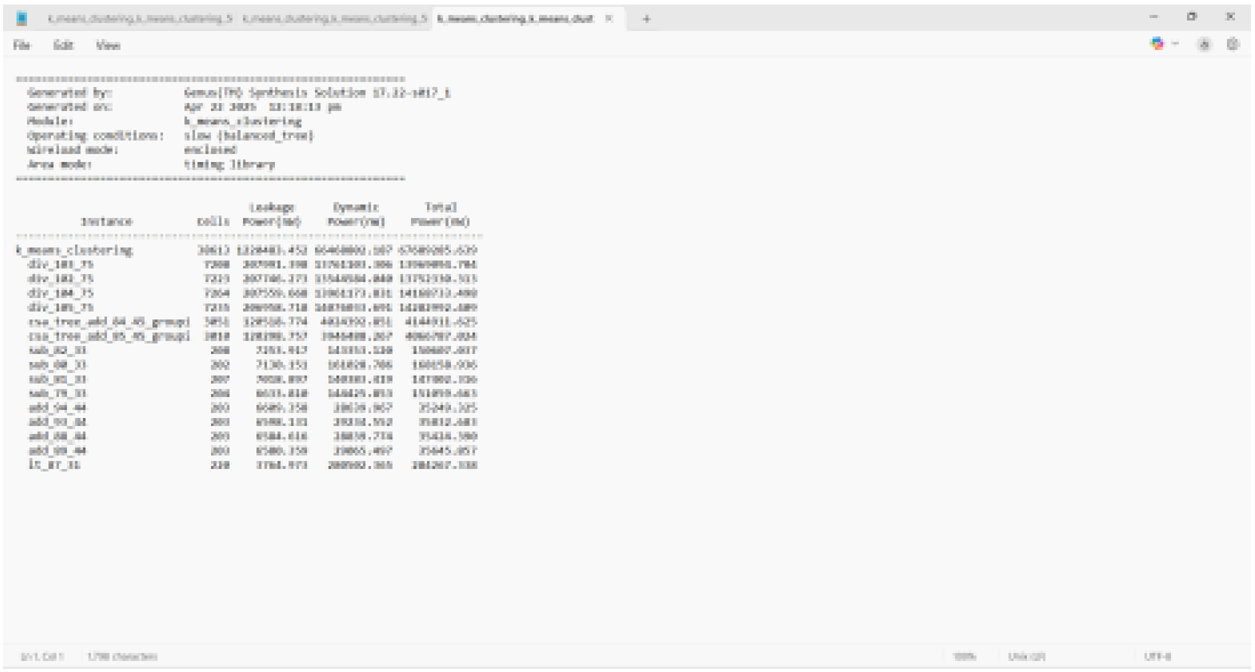
```
OAI31X1           52     314.870    slow
OAI31X2            8      90.818    slow
OAI31XL            6      36.331    slow
OAI32X1           19     129.430    slow
OAI32XL            3      20.436    slow
OR2X1             93     422.350    slow
OR2X2              4      21.193    slow
OR2XL             39     177.315    slow
OR3XL              4      24.221    slow
SDFFQX1            1      24.978    slow
SDFFRHQX1        102    2547.725    slow
SDFFRX2            1      28.762    slow
SDFFSHQX1         32     871.949    slow
XNOR2X1          334    2780.851    slow
XNOR2X2           29     307.301    slow
XNOR2X4           24     272.483    slow
XNOR2XL          498    4146.298    slow
XNOR3X1          202    4586.814    slow
XNOR3XL           20     454.140    slow
XOR2X1            16     133.214    slow
XOR2X2             2      18.366    slow
XOR2X4             4      45.414    slow
XOR2XL           578    4812.370    slow
XOR3X1            21     476.847    slow
XOR3XL             5     113.535    slow
-------------------------------------------
total          38613  212851.958


     Type      Instances     Area     Area %
---------------------------------------------
sequential           542  10075.056     4.3
inverter            8897  23338.255    10.0
buffer               678   3398.696     1.5
logic              28496 196843.931    84.2
physical_cells         0      0.000     0.0
---------------------------------------------
total              38613 212851.958   100.0
```

Figure 7: Cell Report @50



```
============================================================
  Generated by:          Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:          Apr 22 2025  04:34:28 am
  Module:                k_means_clustering
  Operating conditions:  slow (balanced_tree)
  Wireload mode:         enclosed
  Area mode:             timing library
============================================================

                                      Leakage     Dynamic      Total
         Instance             Cells  Power(nW)   Power(nW)   Power(nW)
------------------------------------------------------------------------
k_means_clustering            38256 1195944.982 53094899.257 54290844.239
  div_108_54                   7262  206612.114 13067373.299 13273985.413
  div_109_54                   7232  205655.474 12772540.101 12978195.574
  div_105_54                   7277  205532.054 12335295.285 12540827.339
  div_104_54                   7261  204993.575 12111202.267 12316195.842
  csa_tree_add_81_45_groupi    3068  121511.095     3952.938   125464.034
  csa_tree_add_82_45_groupi    3021  118766.501   315697.544   434464.045
  sub_79_33                     199    6919.975    99155.300   106075.276
  sub_77_33                     209    6827.685    91804.795    98632.480
  sub_78_33                     195    6809.506   102735.827   109545.333
  sub_76_33                     205    6660.969    96515.121   103176.090
  add_90_44                     200    6496.640    29376.299    35872.939
  add_91_44                     200    6479.720    29258.940    35738.660
  add_86_44                     201    6454.772    30236.678    36691.450
  add_85_44                     197    6424.784    28568.219    34993.003
  lt_84_31                      209    3561.799    10194.364    13756.163
```

Figure 8: Power Report @10



```
================================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Apr 23 2025  12:18:13 pm
  Module:                 k_means_clustering
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
================================================================

                                    Leakage      Dynamic       Total
        Instance           Cells   Power(nW)    Power(nW)    Power(nW)
-------------------------------------------------------------------------
k_means_clustering         30613 1220483.452 65460802.187 67589285.639
  div_101_75                7308  307981.398 13761303.386 13969898.784
  div_102_75                7323  307740.373 13544584.840 13752330.515
  div_104_75                7264  307559.668 13961173.831 14160733.490
  div_105_75                7215  206958.718 14076033.691 14282992.409
  csa_tree_add_84_45_groupi 3051  120518.774  4024392.851  4144911.625
  csa_tree_add_95_45_groupi 3018  120298.757  3946408.267  4066707.024
  sub_82_33                  208    7251.917   143353.130   150607.037
  sub_88_33                  202    7130.151   161028.786   168158.936
  sub_85_33                  207    7028.897   140383.419   147082.316
  sub_79_33                  208    6633.810   144425.853   151059.663
  add_94_44                  203    6585.358    28636.967    35249.325
  add_93_44                  203    6598.131    29234.552    35832.683
  add_88_44                  203    6584.616    28859.774    35434.390
  add_89_44                  203    6580.159    29065.497    35645.657
  lt_87_31                   220    3764.973   280502.365   284267.338
```

Figure 9: Power Report @50

Step 1: In the Terminal write the nclaunch to open the tool.

Step 2: Copy the file in the download folder of slow.v from the directory saved in vlog folder.

Step 3: Send the slow.v , synthesis after netlist.v and the testbench .v file in the compiler hdl and the result store in the Worklib file.

Step 4: Open the workLib file and select the module name file in .v format same for testbench .v format and add the elaborater on each .v file.

Step 5: Open the snapshots, select the generated testbench file and open the gui graphics .

Step 6 : In new window, Select the Testbench File which is mentioned in .v format select all its input,output,clock and reset.

Step 7 : Run the testbench and observe the result.

Step 8 : Select all the data variables and sent to the waveform to check the functionality. Step 9: (if needed) If error occur check the verilog code from the scratch and keep the functionality intact.

Note: Keep the timescale format in the testbench as well as in synthesis netlist If all these steps works properly with no error we can now move to Physcial Design on Innovus Tool.

6 Genus Schematic

The Genus Schematic Viewer enables visualization of the synthesized gate-level design, dis playing standard cells, connections, and signal flow. It helps in debugging, understanding logic implementation, and analyzing design hierarchy. Users can trace paths, examine con

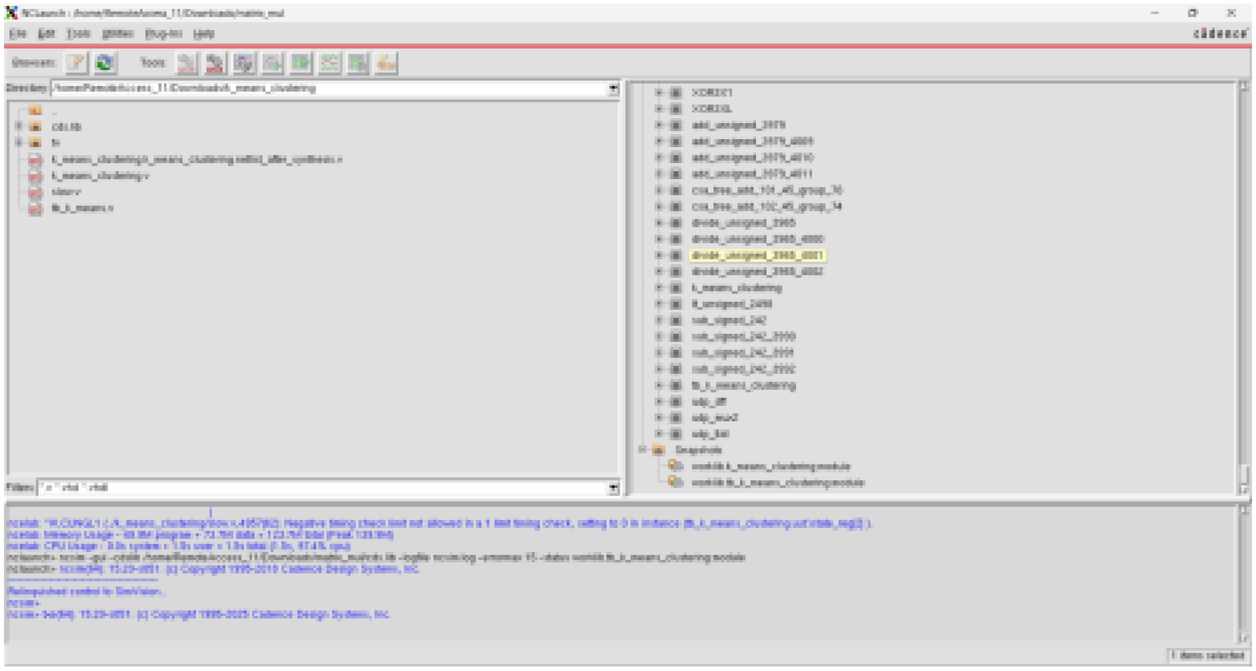


Figure 10: NCLAUNCH Window Setup

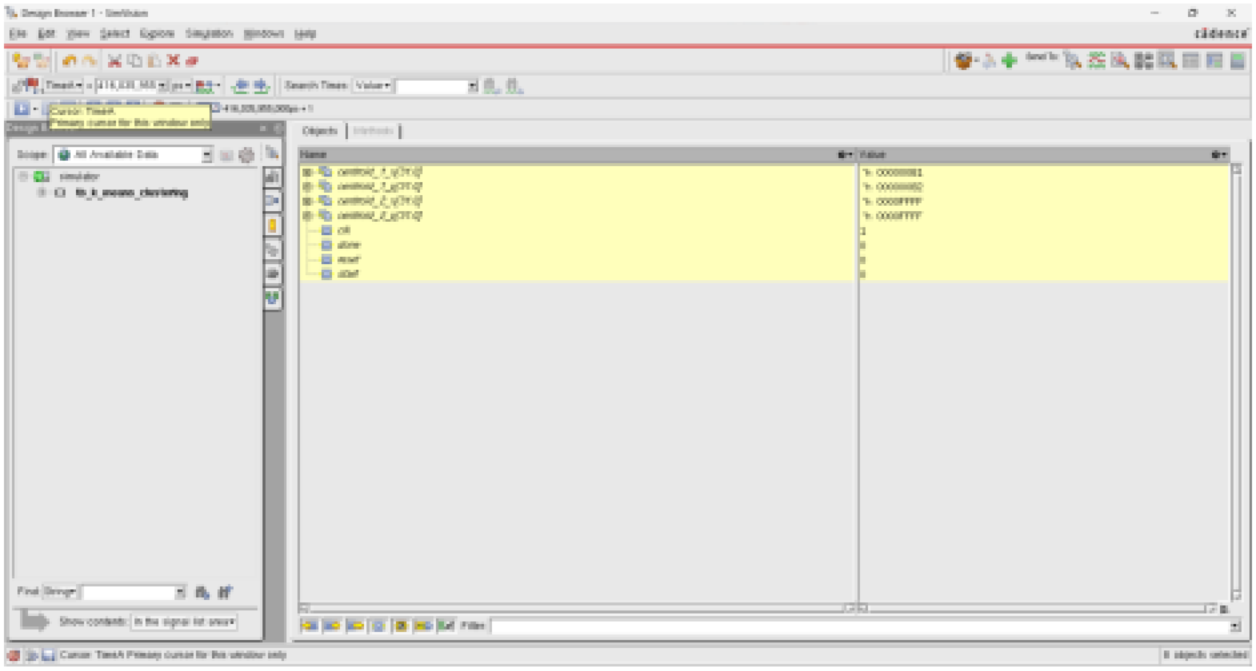


Figure 11: NCLAUNCH Variables Selection

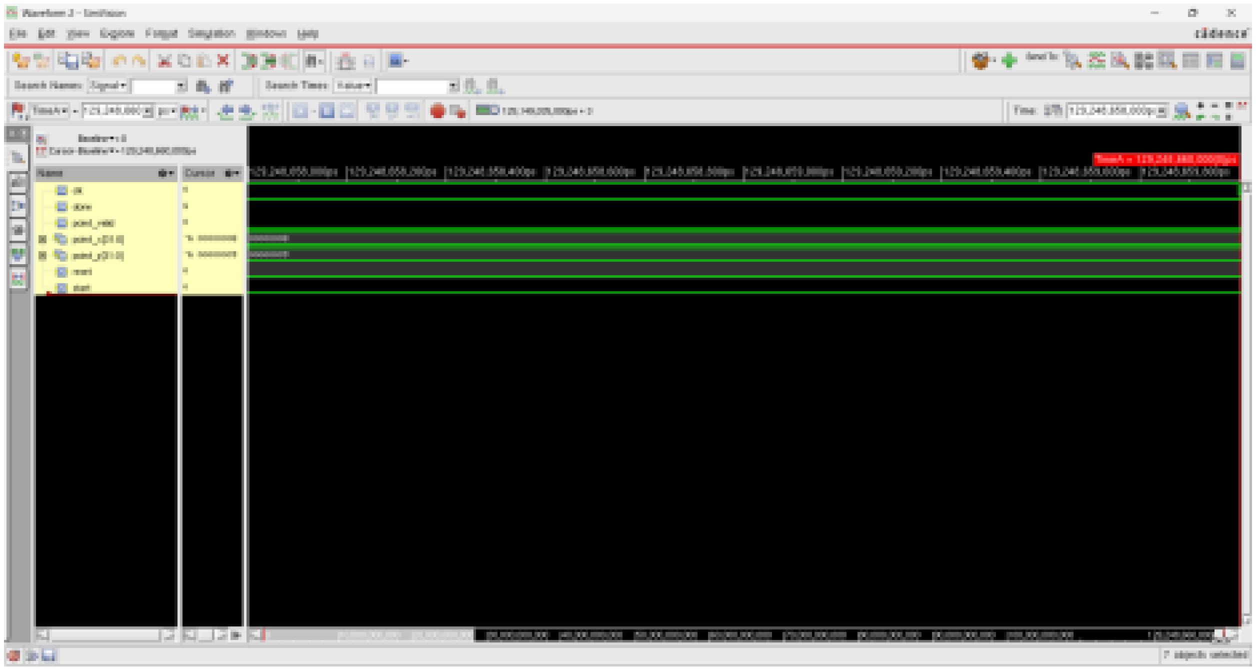


Figure 12: Final Testing Simulation

trol logic, and ensure correctness before moving to physical design, enhancing overall design verification efficiency.

7 Physcial Design

Physical design is about placing the gates and connecting them with wires on a chip. It includes steps like floorplanning, placement, clock tree synthesis (CTS), routing, and signoff checks. The goal is to make sure the design fits on the chip, works fast, uses less power, and meets all timing rules before manufacturing.

Defines the layout area, places major blocks (like memories, macros), and sets up power planning. It's like deciding where rooms go before building a house.

Standard cells from the netlist are placed in rows to minimize wire length and meet timing. No routing yet—just positioning the gates efficiently.

Builds a balanced tree of buffers/inverters to evenly distribute the clock signal to all flip-flops. This helps reduce clock skew and timing problems.

Wires are drawn to connect the placed cells based on the netlist connections. The goal is to avoid congestion, shorts, and delays.

Step 1 : In the terminal type the command innovus to open the tool,then a new window of cadence open.

Step 2 : Go to the file, open the design, now fill all the essential files ,LEF files , netkist after synthesis.v files, and in MMMC add the max Timing with slow lib and min timing with fast lib, max delay with max timing and vice versa,add the setup and hold condition,in last all the netlist and design floor occur on the screen. The format will be of .view and .global. Step 3 : Open the specify floorplan fill the necessary information like core utilization and

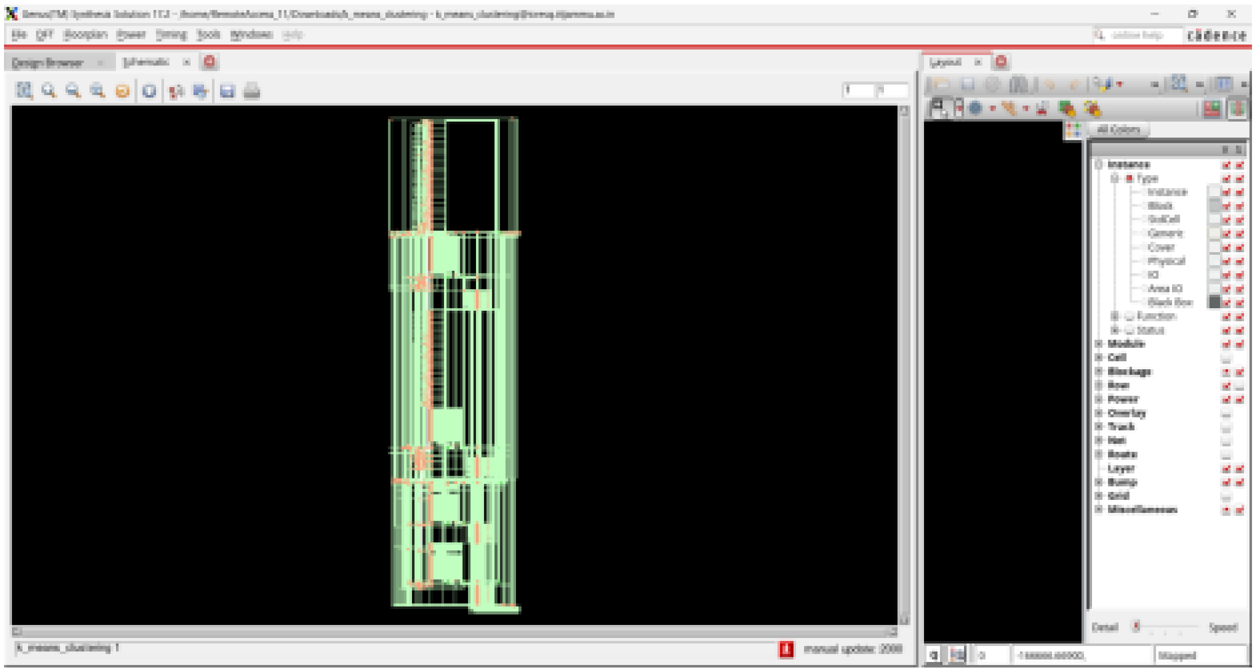


Figure 13: Genus K Mean Clustering Schematic

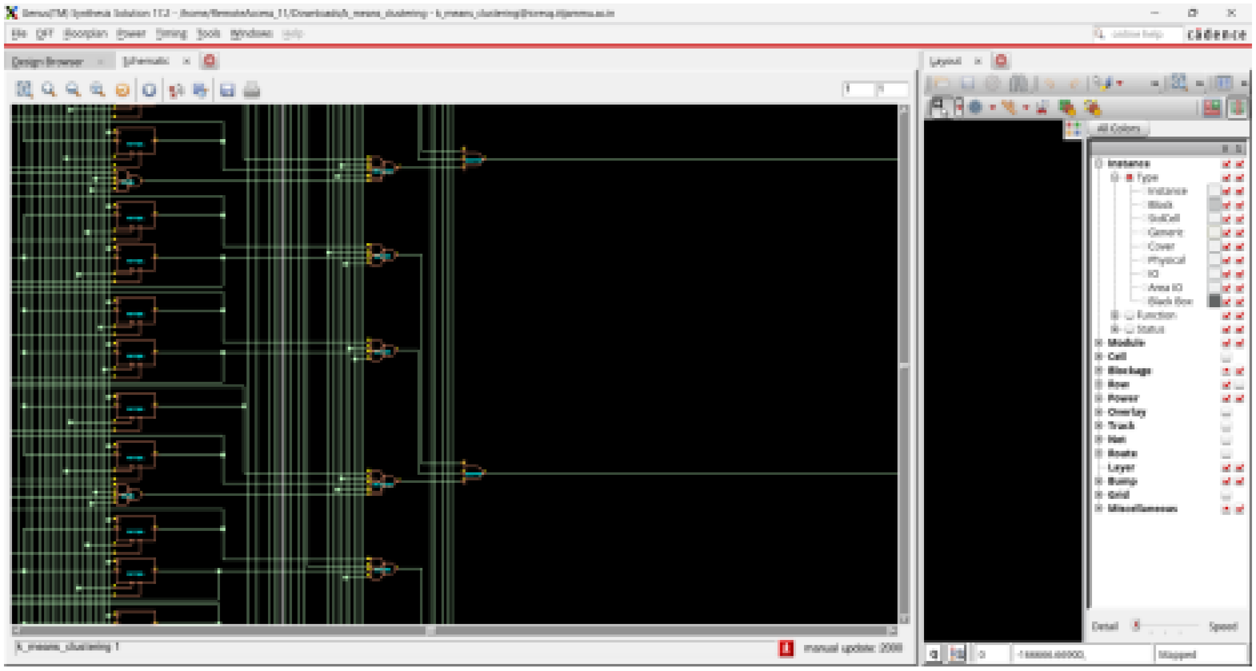


Figure 14: Std. Cell

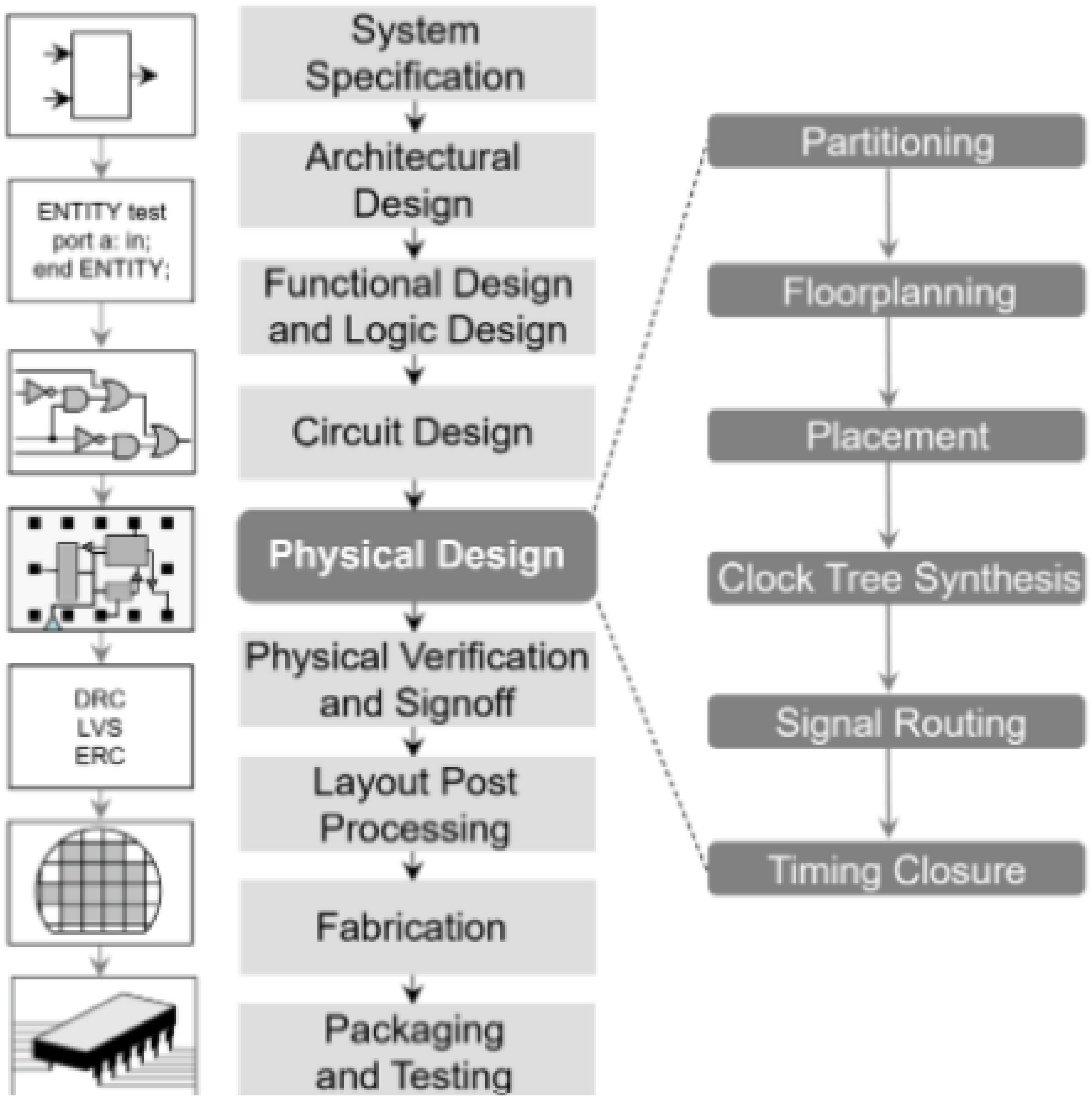


Figure 15: Flow Diagram

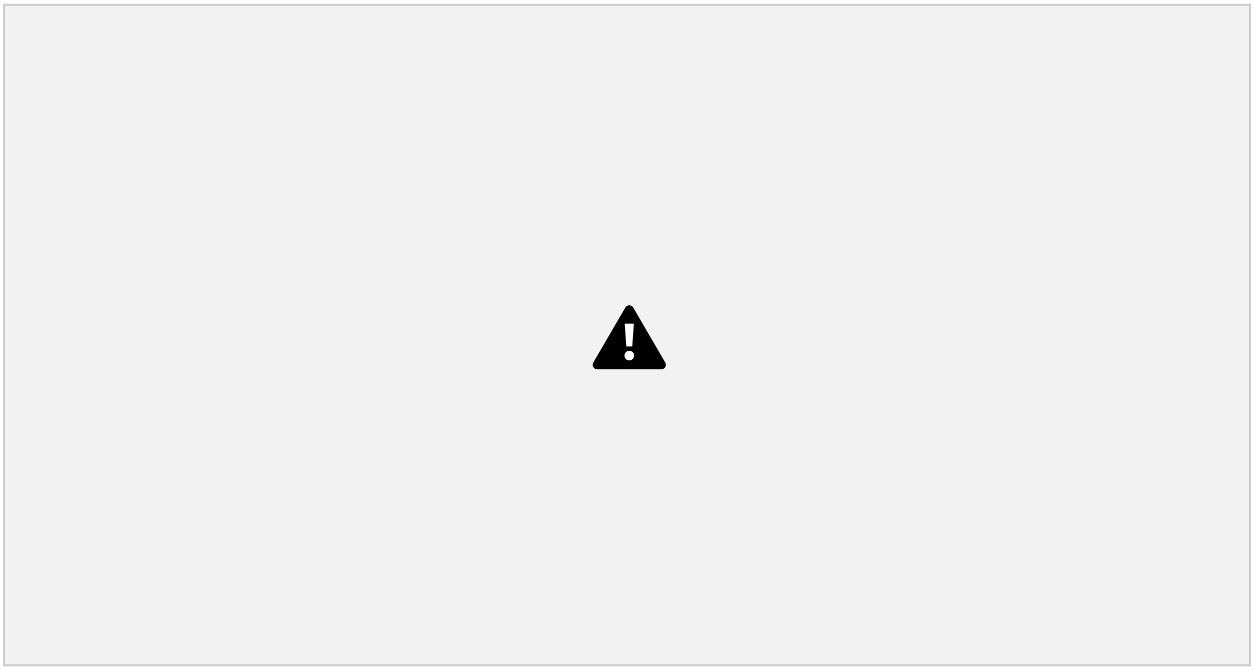


Figure 16: Import the Design

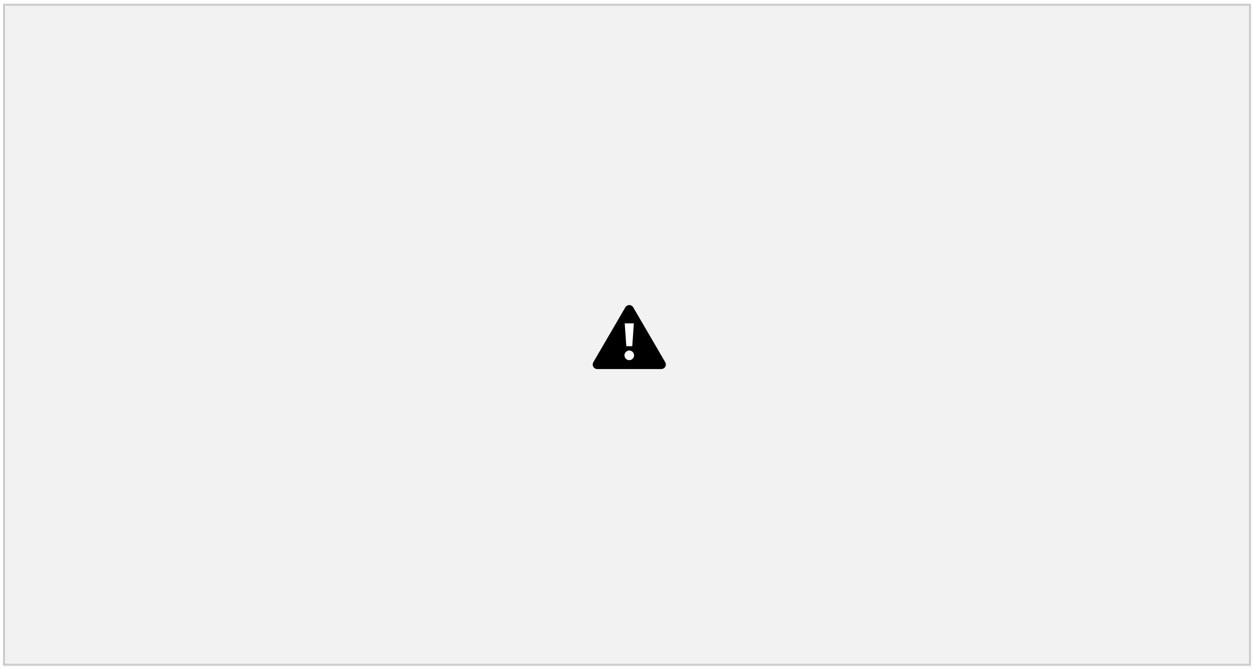


Figure 17: Specify The Floorplan

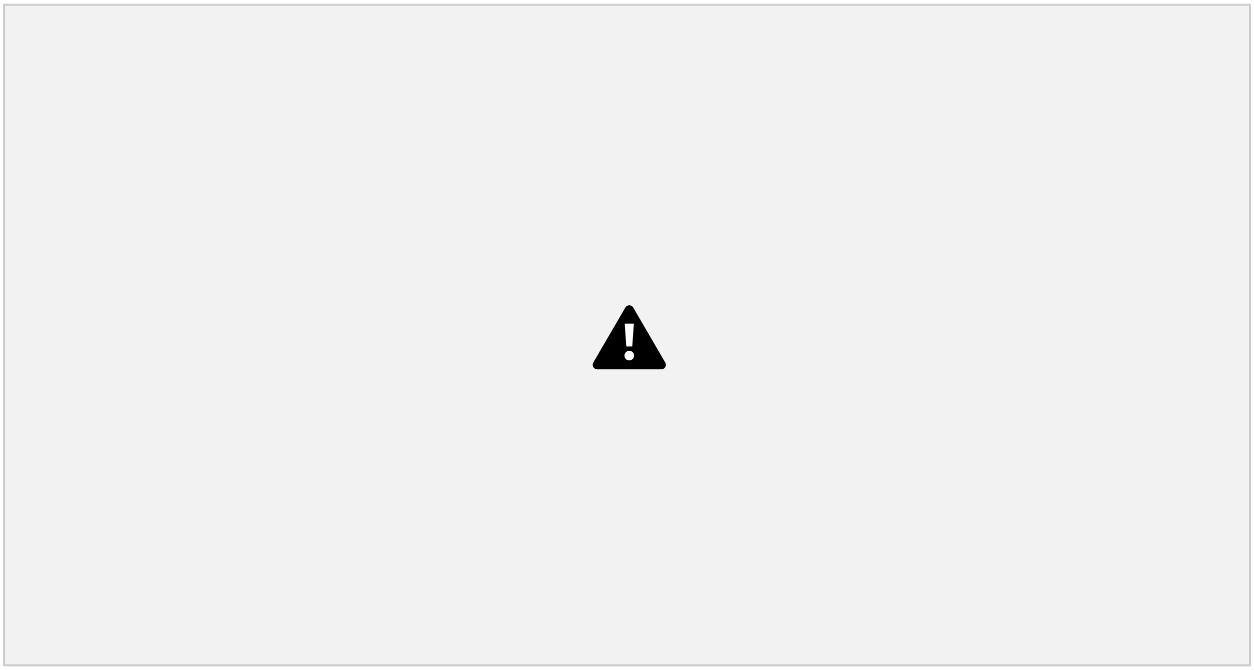


Figure 18: Power Planning

add the aspect ratio and core to IO boundary to 10,click ok. Step 4 :Do the power planning add the strips, rings with appropriate metals selection both horizontal and vertical with high metals.

Step 5 : Add the nets,by clicking on route and then special route and click on all metal layers.

Step 6: Go to place,physical cell, add encp. select the filler and add later the well tap instance with appropriate setting to rows to get the design.

Step 7 : Again Go to place,std. cell, then place io pins and click on ok, all the std. cell come into the die are.

Step 8 : In digital analysis, Go to report timing , preCTS to setup all information occur on the screen, then same will done for the hold .

Step 9: Go to the Clock Debugger click on CTS and apply ,path will be generated and save the design with .enc format.

Step 10:Go to Route ,Nano Route ,route add the timing driven add congestion click on terminal to get the result.

Step 11 : Go to placement add the physical cell add the filler(Select all filters) the fillers get added on the vacant space.

Step 12: Go to ECO,optimization Design post route add ok.On window we get the optimized design.

Step 13: Go to file save the design in GDS2 with stream.out format and our physcial design is completed.

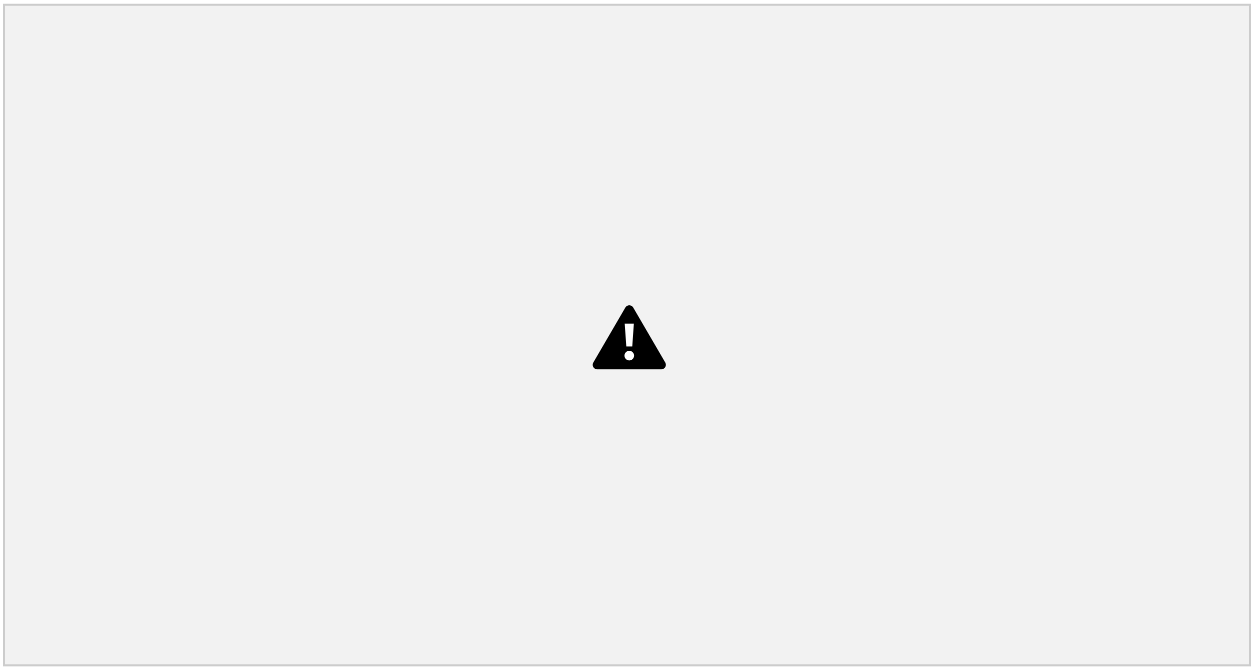


Figure 19: Route Method

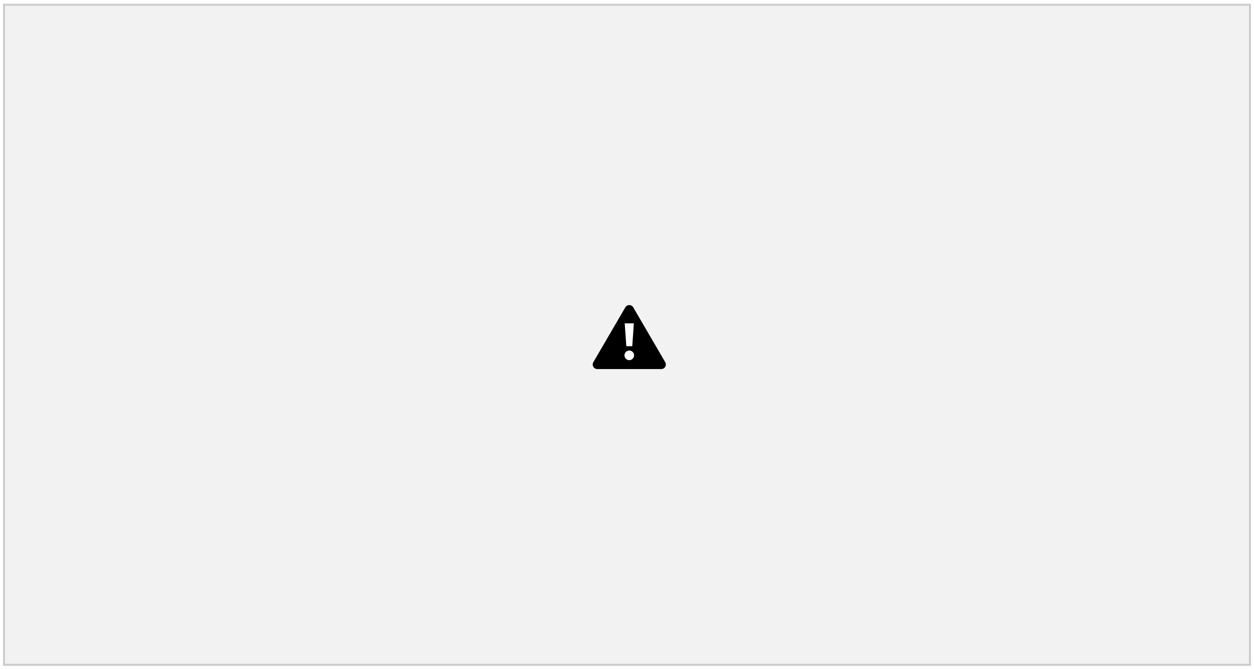


Figure 20: PLACING STD. CELL

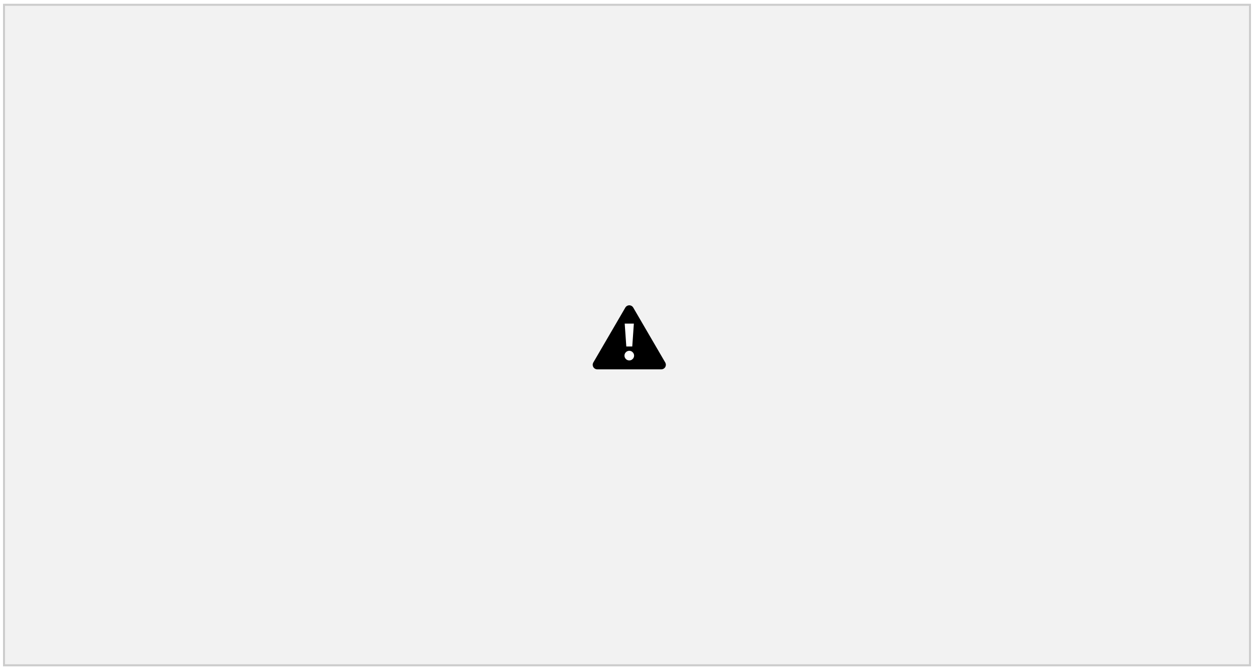


Figure 21: pre CTS hold

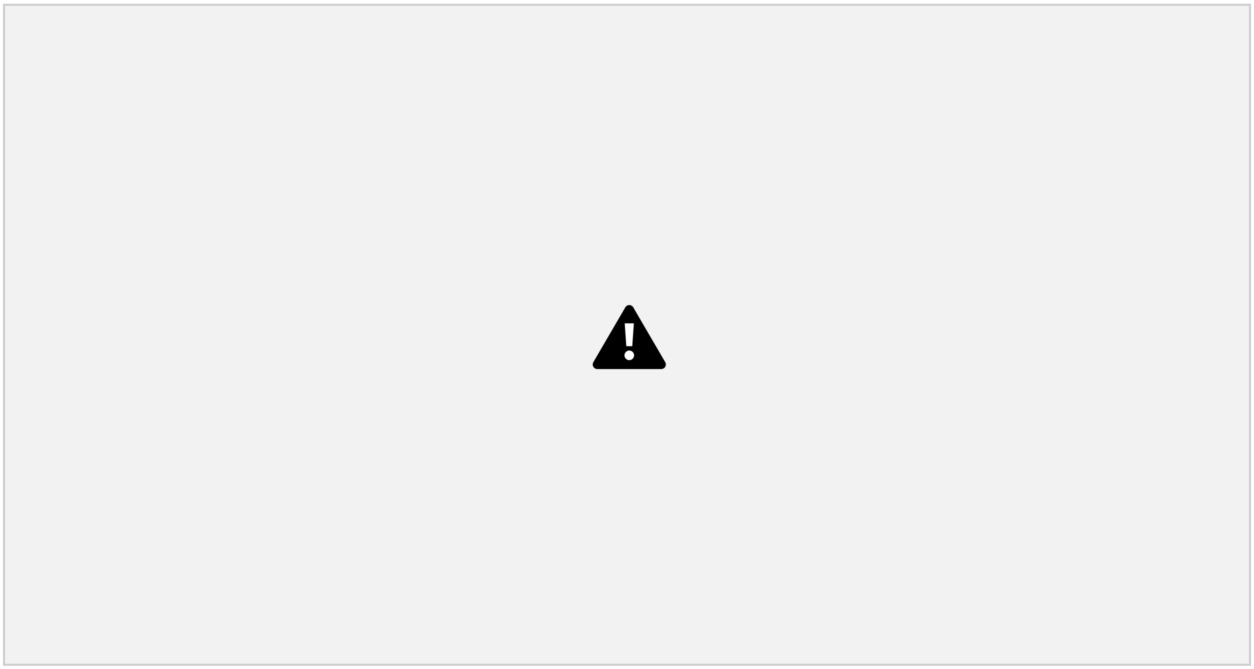


Figure 22: Clock Debugger

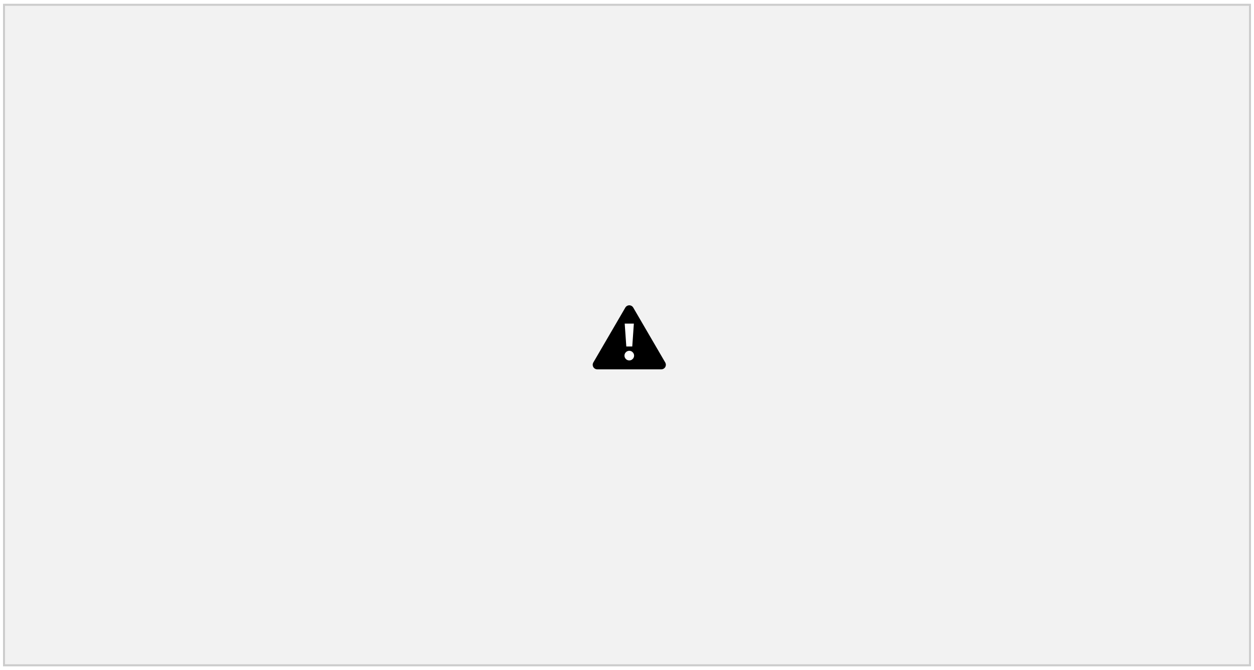


Figure 23: Clock Path

8 Conclusion

The integration of K-Means clustering in the ASIC design flow proved helpful in optimizing performance and managing design complexity. By grouping similar logic elements during synthesis and physical design, tools like Genus and Innovus benefited from better placement, reduced wire length, and improved timing. This method helps in lowering power consumption and increasing design efficiency, especially for large-scale VLSI circuits. Overall, using K Means clustering supports better design quality and faster convergence in the flow, making it a useful approach for modern chip design and implementation.

Figure 24: Clock Selector

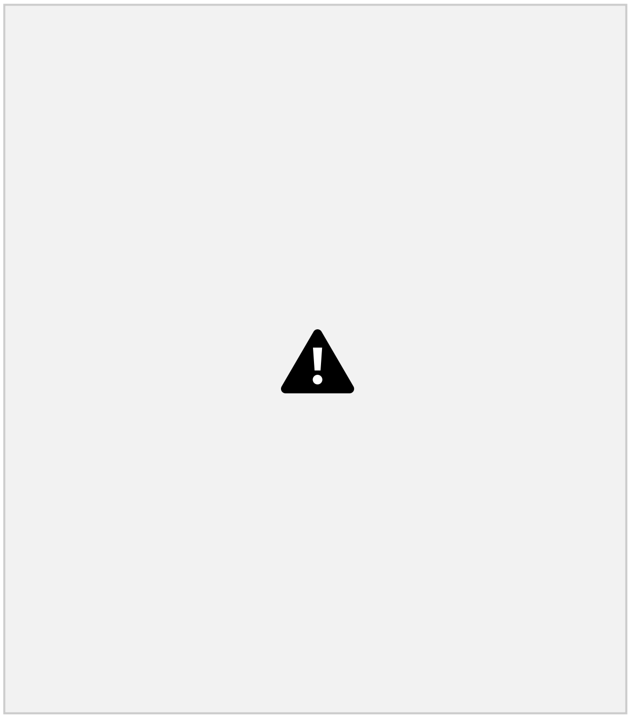


Figure 25: Power Driven

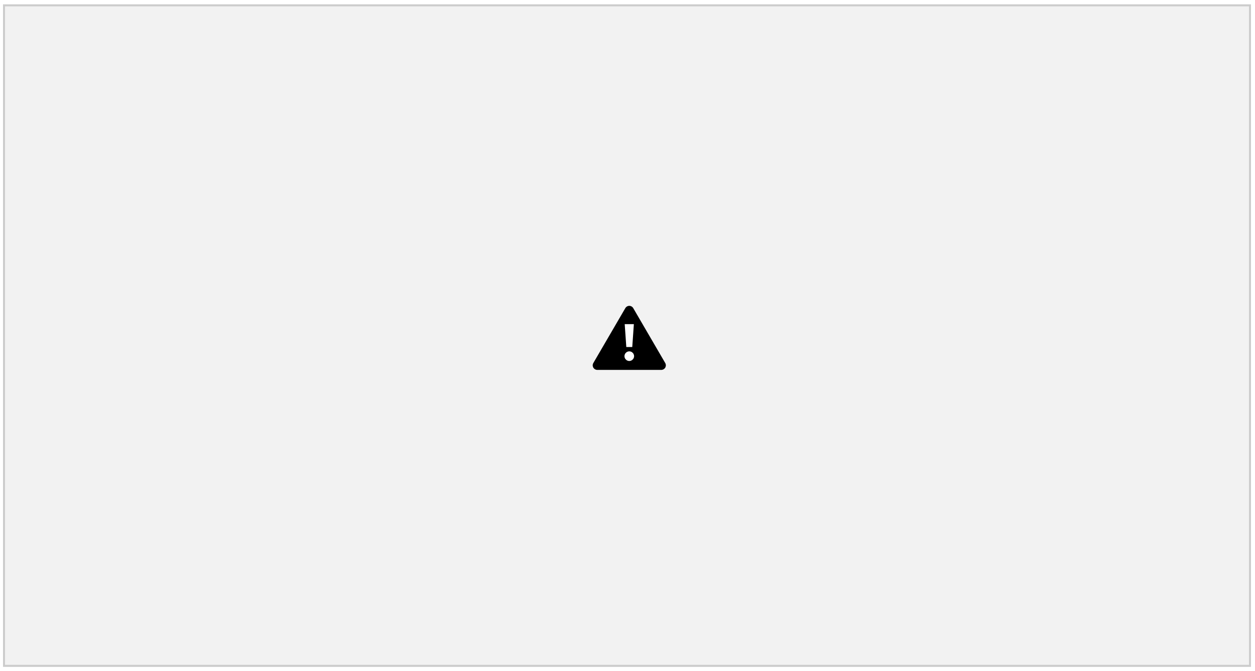


Figure 26: Time Detailed Routing

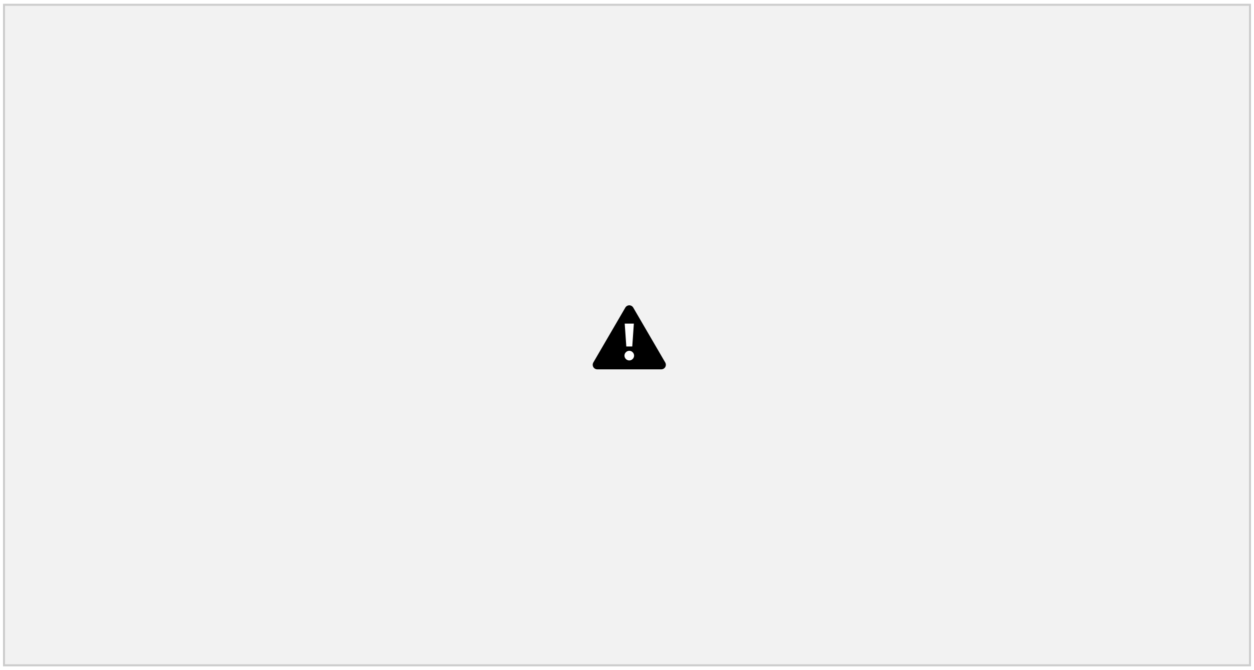


Figure 27: Adding Fillers

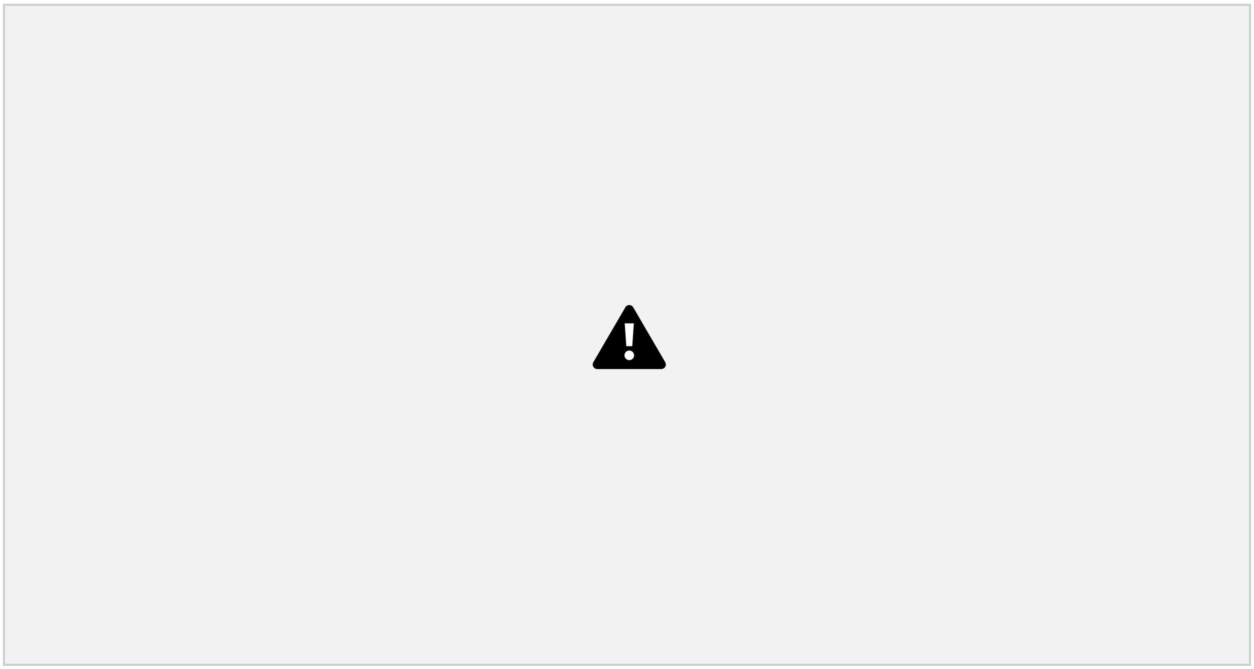


Figure 28: Pre Opt std cell

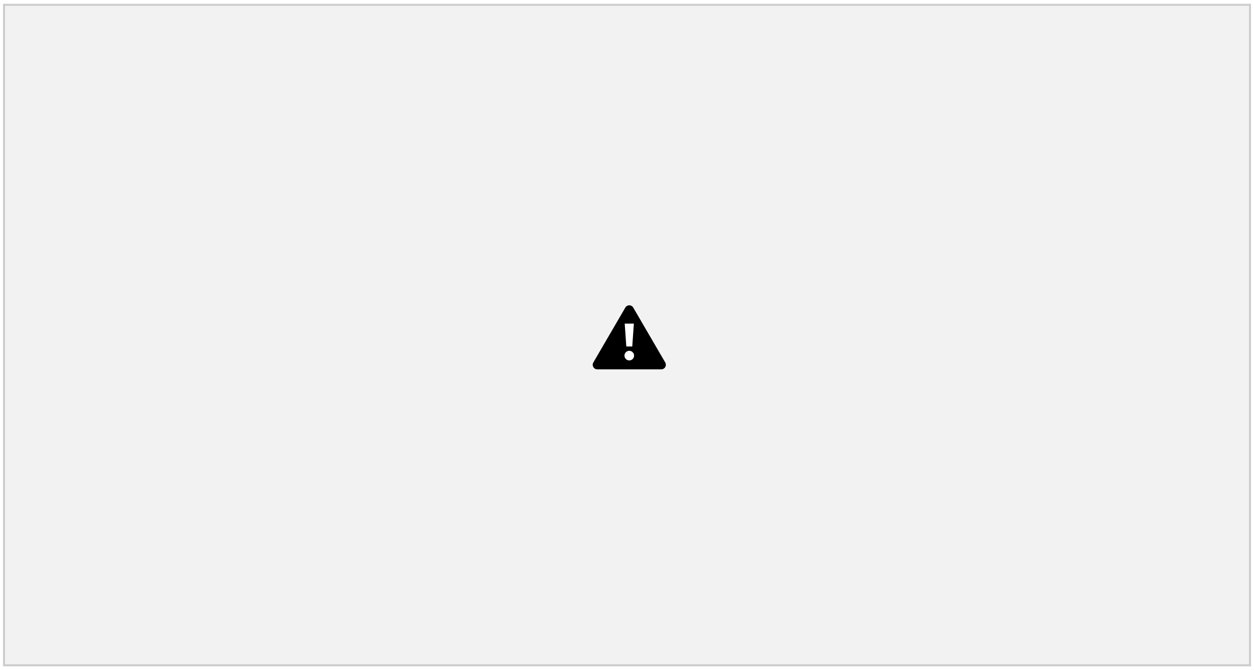


Figure 29: Optimzing Step

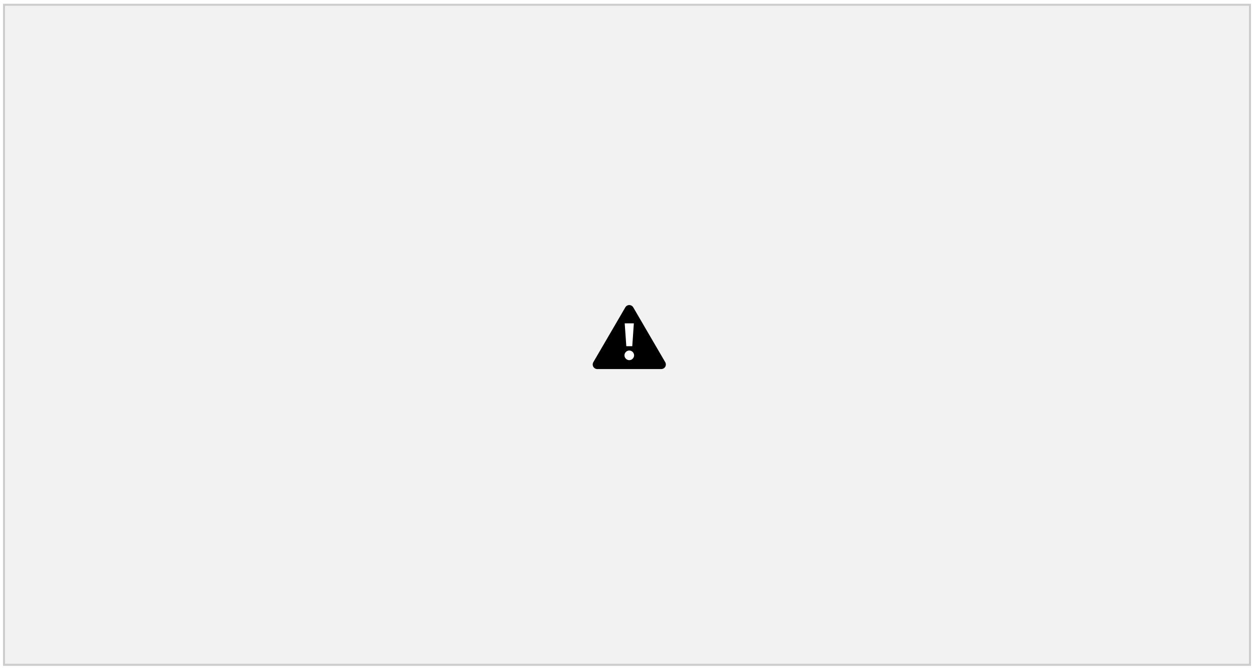


Figure 30: Optimized Filling

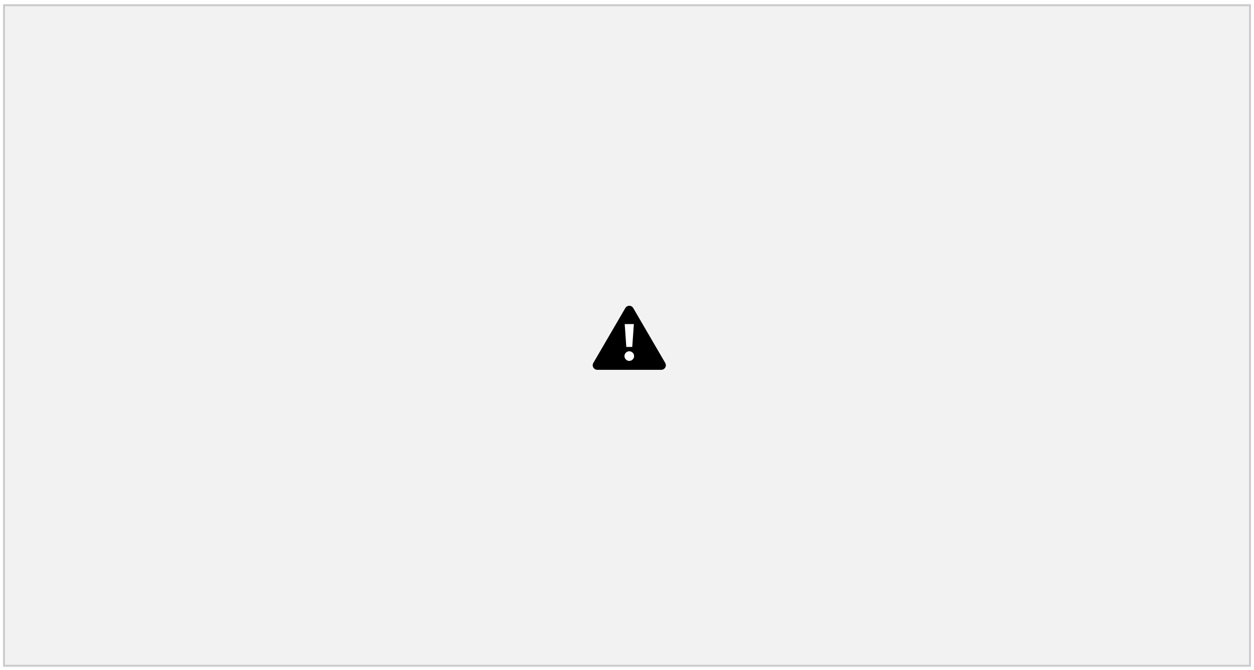


Figure 31: Final Design