Sharif University of Technology
Department of Computer Engineering

Advanced VLSI Design Course Project
Physical Design and Layout Flow Using Cadence for Synthesized Digital Circuits

Mahbod Afarin
December 2015

1- Introduction and Problem Definition

In project 2, we used the Design Compiler (DC) tool to convert high-level code into a **netlist**, ultimately producing a **gate-level circuit**. Essentially, we created a mapping between the **technology cells** and the **circuit we had**. In this project, we want to provide our circuit to the **SOC Encounter** software so that it performs **Placement & Routing (P&R)**. The output can then be sent to a **fabrication facility**, which can manufacture the IC for us. In this task, we will **not optimize the circuit using Desing Compiler**, because optimization may lead to some gates being removed, and as a result, the final circuit might contain wires that are **not connected anywhere**.

2- Project Steps

It is important to always open **SOC Encounter** from the **directory of your design**.

We navigate to the appropriate path and provide the **netlist**, which was generated as an output from DC. Then, we enter the **top module name**, and we also add the **.sdk file**, which defines the timing constraints.

The next file is the **.lef file**, which contains the **physical technology** information. This tells the software things like how many layers the standard cells have, the pitch, and so on.

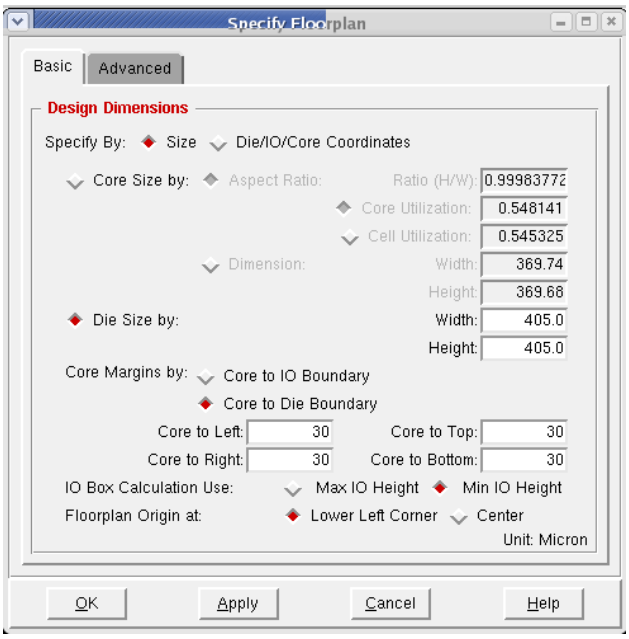
For the **LEF file**, we add **two files**:

- One is the **standard cell LEF file**
- The other is the **antenna effect file**

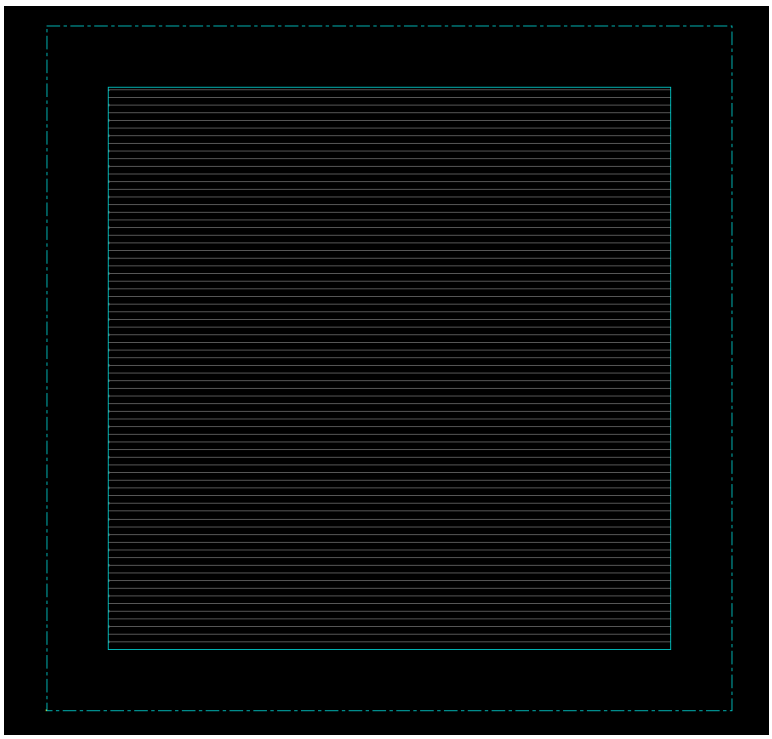
The **antenna effect** refers to the issue that arises when routing becomes dense: each wire behaves like an antenna, and its capacitance can introduce **noise** into the circuit. To manage this, we use the **antenna library**, which contains antenna-related specs for each cell.

Next, under the **Advanced** tab and the **Power** section, we define the **global VDD and GND** for the circuit. All standard cells must have their VDD and VSS specified. Since VDD and GND come from outside the chip, we must explicitly define the global power and ground for our design.

Then, as shown in the figure, we go to **Specify Floorplan**. We activate the option **"Die size by"** and assign an area larger than the default. The default area is minimal, so we increase it to ensure sufficient space.



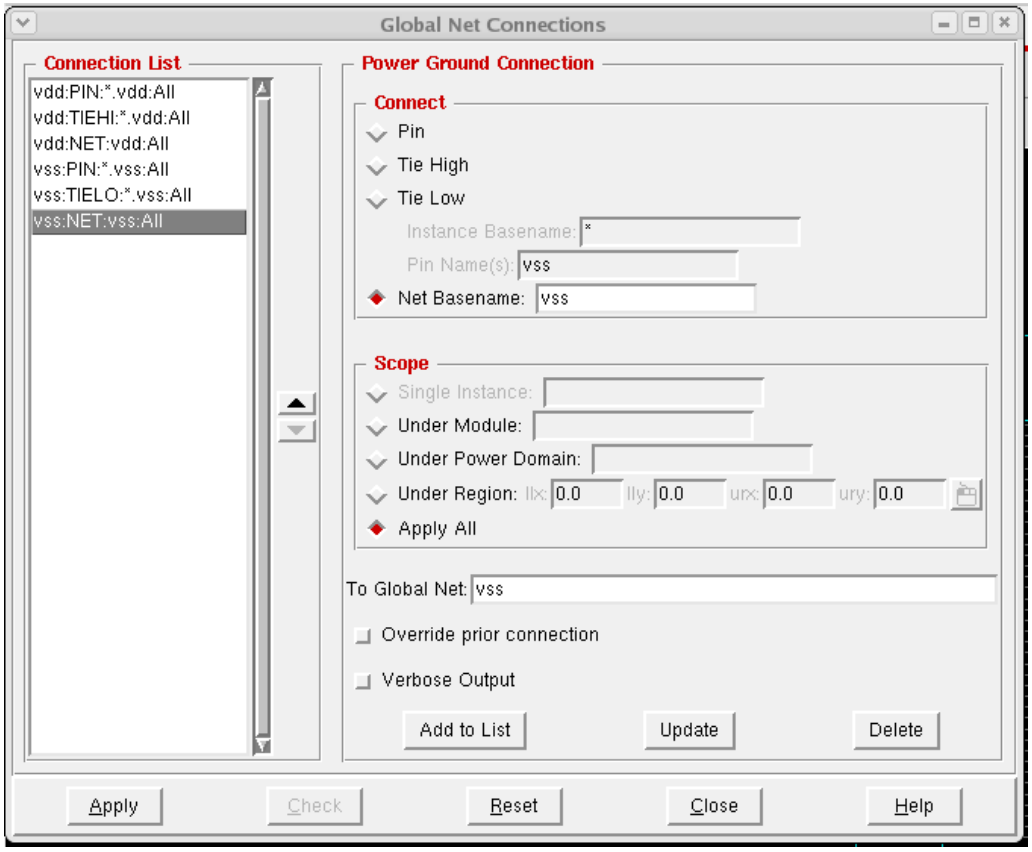
Next, to connect the core to **VSS** and **VDD**, two complete **power rings** (one for VSS and one for VDD) are drawn around the core. The existing VDD and VSS pins in the core are then connected to these rings. This approach helps to **reduce power consumption**, **increase noise immunity**, and **optimize routing**. Therefore, in the **"core to left"** and **"core to right"** sections, we specify a spacing value to provide room for the power rings. The result is as shown in the figure below.



As shown in the figure above, a **spacing** has been created between the **core** and the **die**. Then, as shown in the next figure, we go to the **"Connect Global Net"** section and connect the **power** and **ground** cells to the **global VDD and GND**.

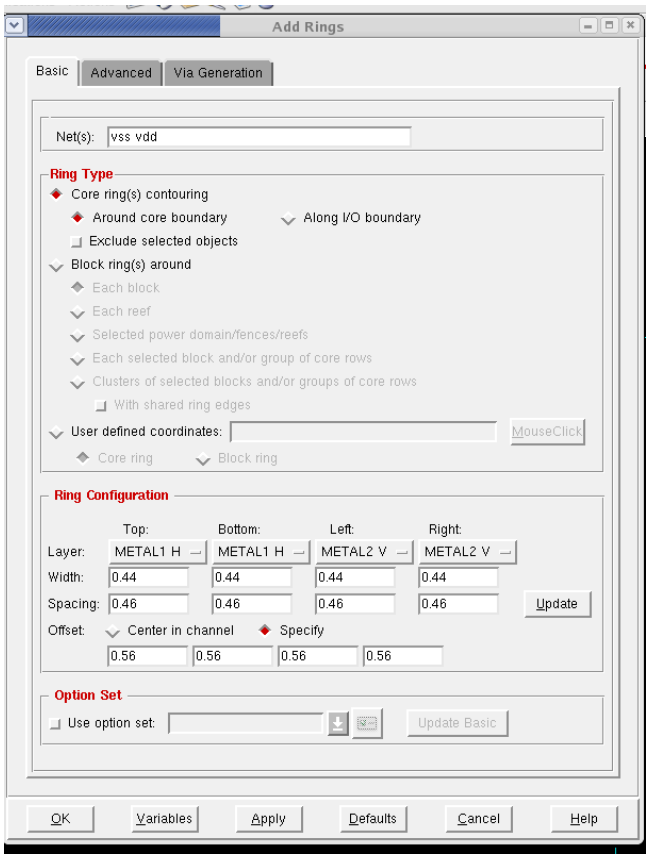
The **tie-high** concept means that in digital circuits, when we want to assign a logic high (‘1’) to a node, we **do not connect it directly to VDD**. This is because if **noise** appears on VDD, it can affect the connected node as well. To prevent this, two special cells called **tie-high** and **tie-low** are used:

- **tie-high** generates a logic high ('1') that is **resistant to noise**
- **tie-low** generates a logic low ('0') that is **also noise-resistant**

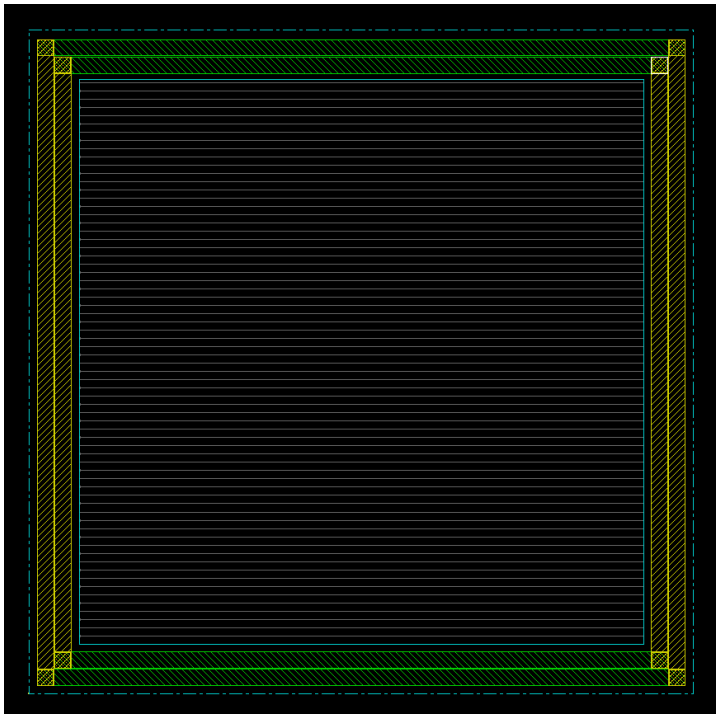


hen, we go to the path: **Power → Power Planning → Add Ring**.

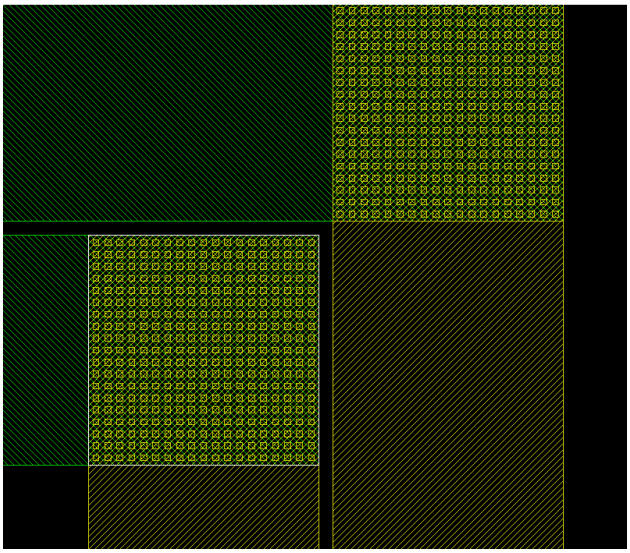
Power rings are usually implemented using **intermediate metal layers**, as they need to **reach all areas** of the chip. We make the following changes as shown in the figure below.



In the figure below, it can be seen that our power rings have been added.



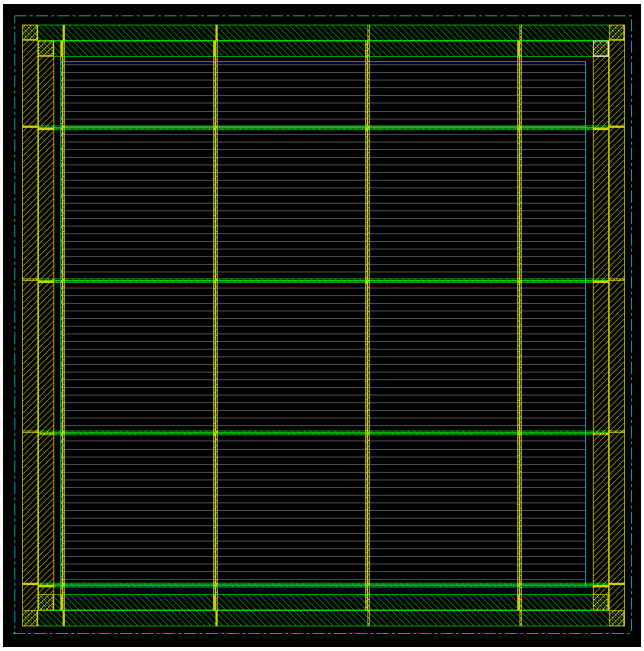
By zooming in, the connections can also be observed.



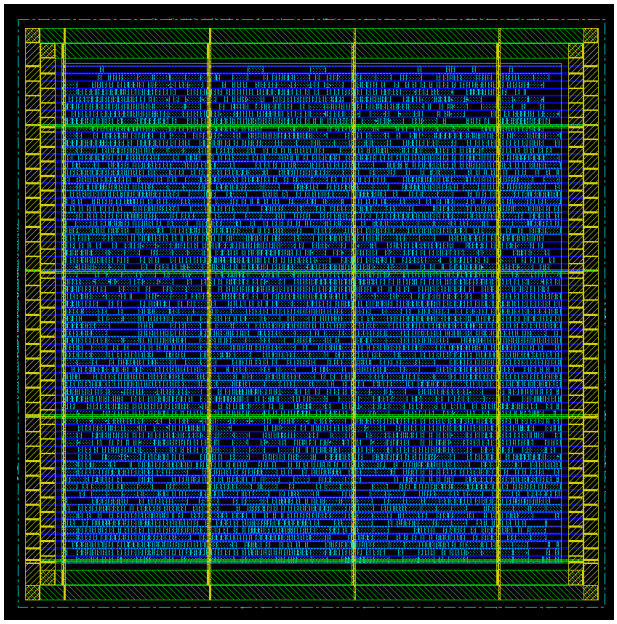
Next, we need to **add strips** to the circuit. This means adding another layer of metal, which will be connected to **VDD and GND**.

If there is a cell located in the middle of the core and we want to connect it to VDD or GND, it must connect through these **strips**. This is illustrated in the figure below.

Navigate to:
Power → Power Planning → Add Strip



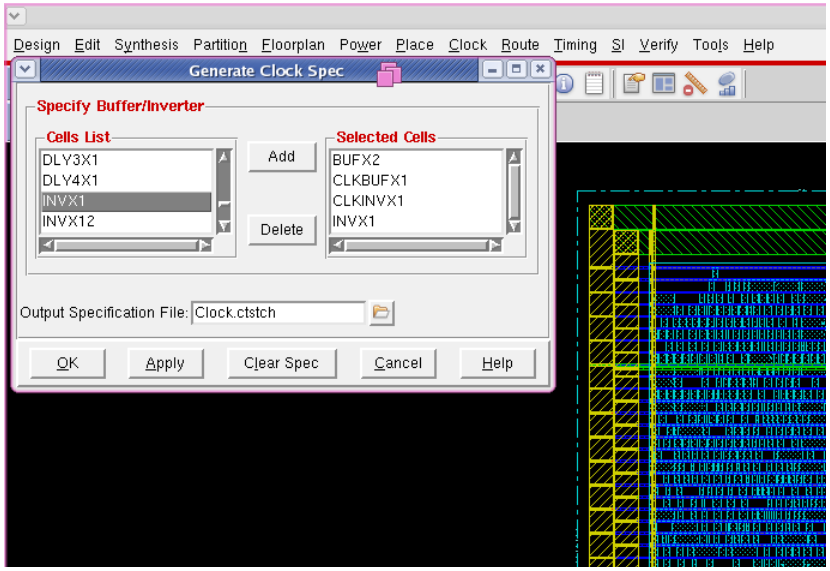
Next, we perform a special route. The output will look like the figure shown below.



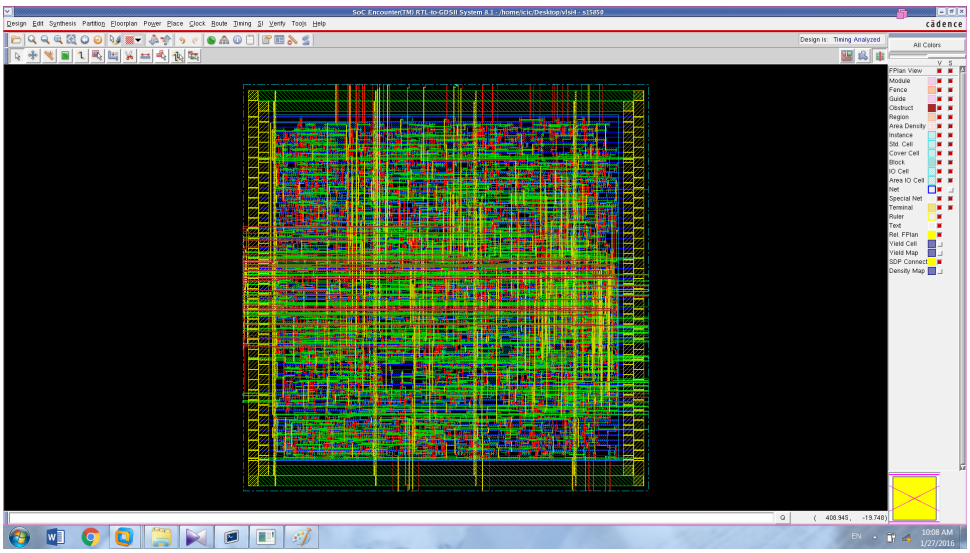
In the next step, the **final routing** and **metal filler insertion** are performed to complete the layout of the design.

After placing the components, it's time for **routing**.

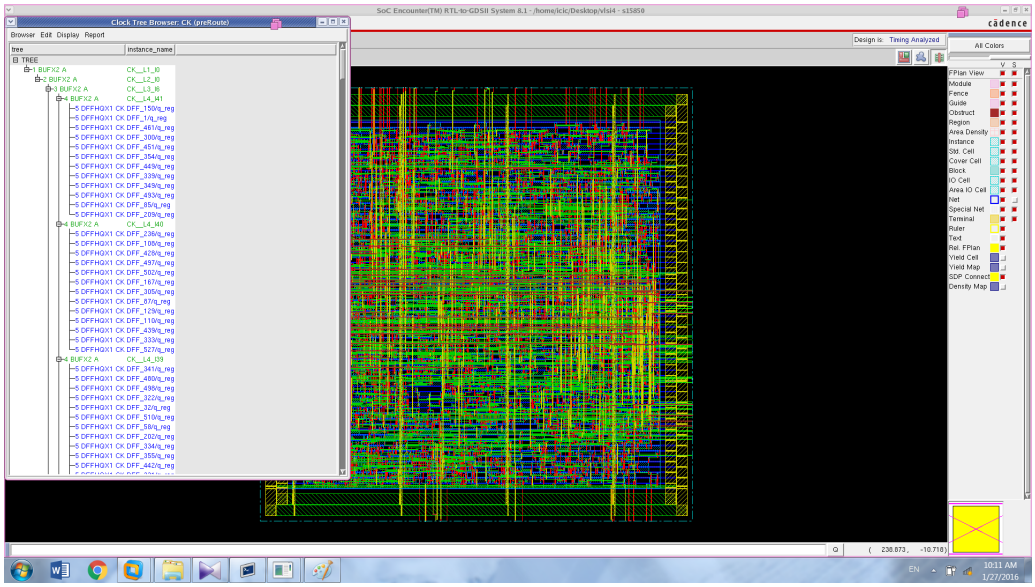
Before routing the **clock signal (clk)**, we need to add certain cells to the circuit to **prevent clock skew**, **voltage drop on the clock**, and other related issues.



After that, the clk, VDD, and VSS signals are connected to the cells.



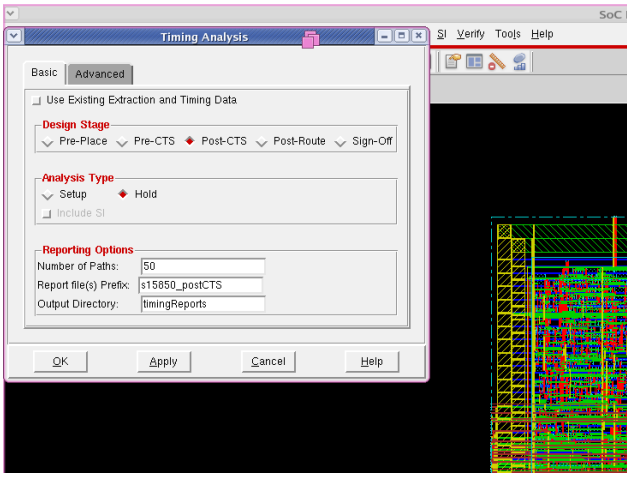
We can view the **clock signal routing path** through this window:



Now, we analyze the setup time and hold time for the clk signal. There should be no negative timing values in the output.

```
------------------------------------------------------------
         timeDesign Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|           WNS (ns):| 14.392  | 14.798  | 14.392  |   N/A   |   N/A   |   N/A   |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |   N/A   |   N/A   |   N/A   |
|    Violating Paths:|    0    |    0    |    0    |   N/A   |   N/A   |   N/A   |
|         All Paths:|   534   |   512   |   422   |   N/A   |   N/A   |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |             Real              |       Total      |
|      DRVs      +---------------+---------------+------------------|
|                |Nr nets(terms)| Worst Vio      |Nr nets(terms)|
+----------------+---------------+---------------+------------------+
|   max_cap      |    2 (2)      |    -0.043     |    2 (2)     |
|   max_tran     |    0 (0)      |     0.000     |    0 (0)     |
|   max_fanout   |    0 (0)      |       0       |    0 (0)     |
+----------------+---------------+---------------+------------------+

Density: 63.303%
Routing Overflow: 0.00% H and 1.21% V
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 1.39 sec
Total Real time: 2.0 sec
Total Memory Usage: 293.742188 Mbytes
encounter 1>
```

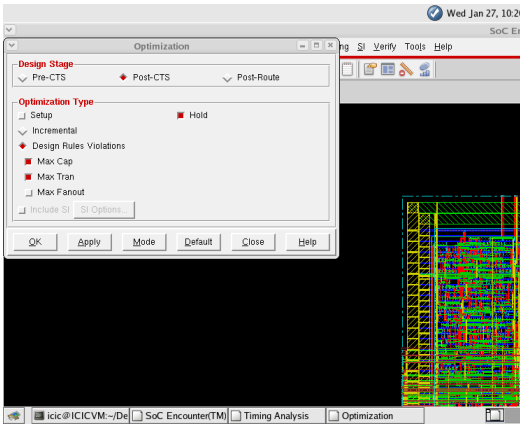
As shown in the image above, there are no negative values for the setup time. Now, let's examine the hold time:



In the output, we will see that negative values are present, which means the hold time is violated.

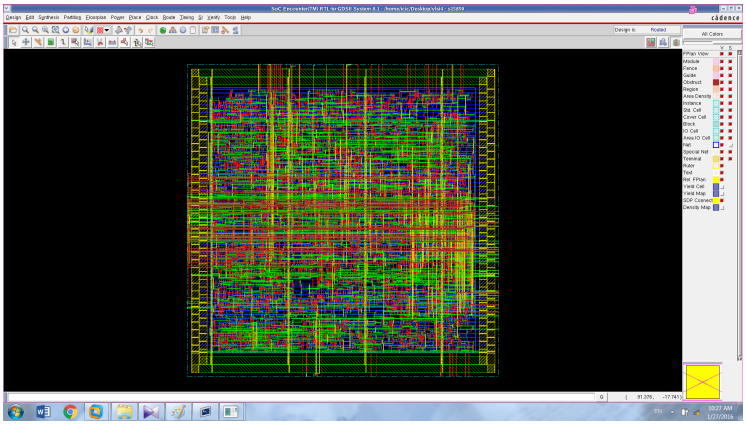
```
------------------------------------------------------------
         timeDesign Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|           WNS (ns):| -0.817  |  0.224  | -0.817  |   N/A   |   N/A   |   N/A   |
|           TNS (ns):|-118.806 |  0.000  |-118.806 |   N/A   |   N/A   |   N/A   |
|    Violating Paths:|   297   |    0    |   297   |   N/A   |   N/A   |   N/A   |
|         All Paths:|   534   |   512   |   422   |   N/A   |   N/A   |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+

Density: 63.303%
Routing Overflow: 0.00% H and 1.21% V
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 1.36 sec
Total Real time: 1.0 sec
Total Memory Usage: 293.742188 Mbytes
encounter 1>
```

In this case, we need to use optimization techniques to try to improve the hold time.

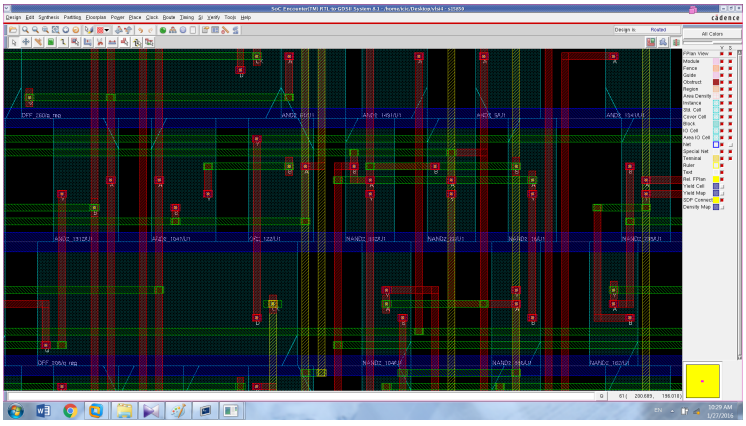


We repeat this process until the hold time becomes positive.

After that, we proceed with routing the remaining signals in the circuit until we reach the layout shown below.

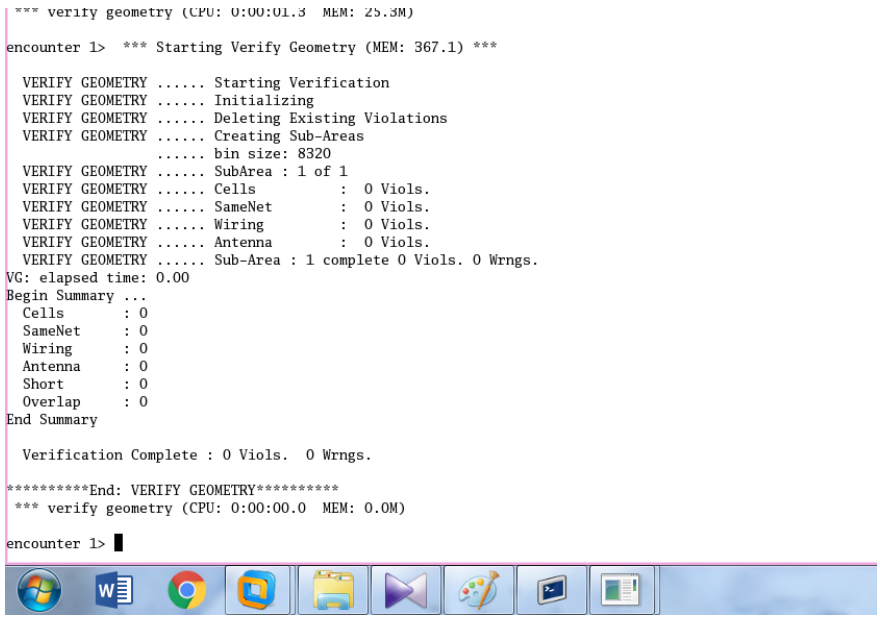


A closer view of the routing layout is shown below:

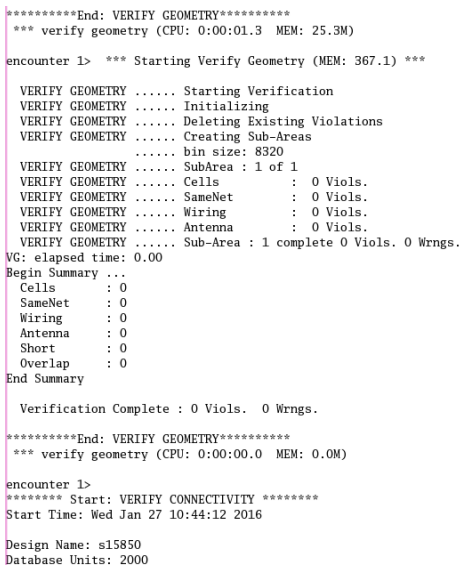


In the next step, we move on to verifying geometry and connectivity, with the positive results of each step shown below.

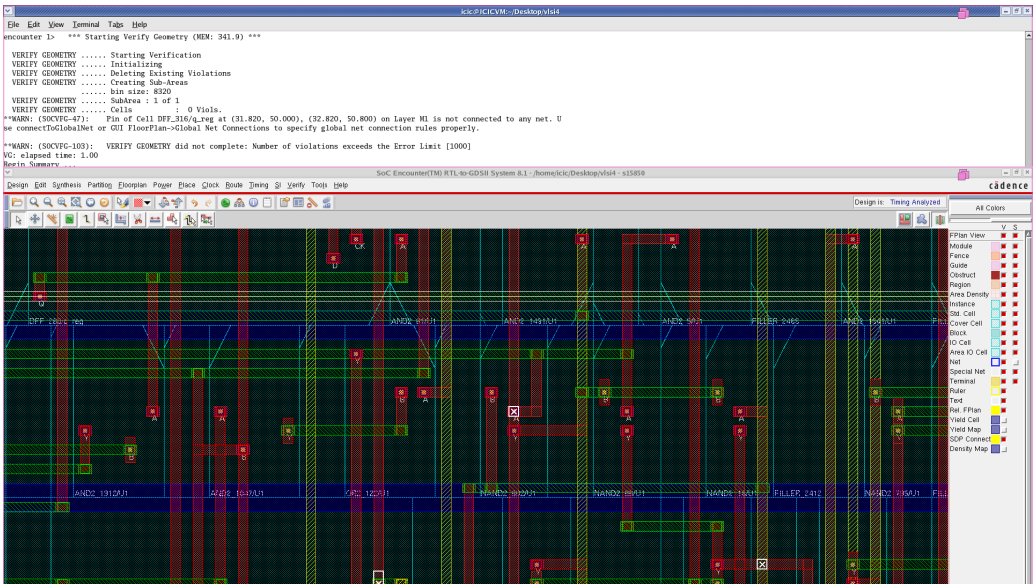
Geometry:



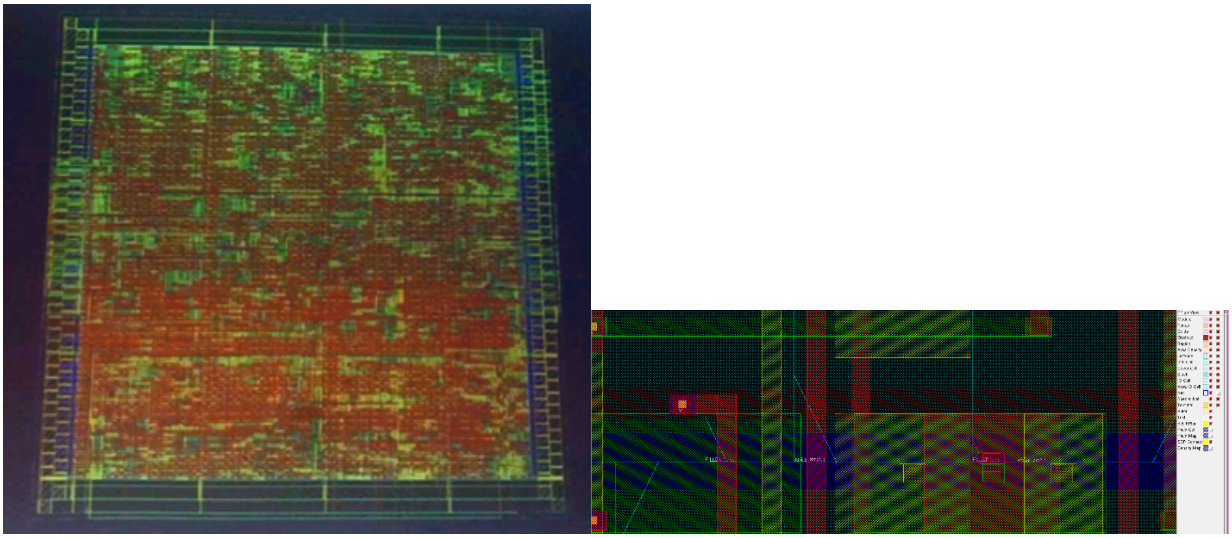
Connectivity:



After this step, we proceed to **Add Filler**, which fills the gaps between cells to make the layout **manufacturable by the fabrication plant**. After performing this step, the **gaps between cells are filled** as shown below:



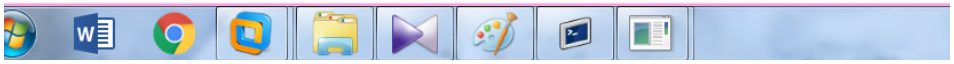
Now, we can perform the **metal fill** step: In this step, **all metal layers** are selected, and the **empty spaces** are filled with the corresponding metal. We then arrive at the layout shown below:



And then we **save** the design. **Note:** After several steps, the saved files are available and have been attached. Once the layout is finalized, it must undergo **timing analysis** to verify the correctness of the placement.

In the next step, we **remove the antenna effect** using the **ANTENNATF diode cell**, and then we re-check the **timing analysis**. We observe that there are **no negative values** for either **hold** or **setup** time.

```
------------------------------------------------------
       timeDesign Summary
------------------------------------------------------

------------------+---------+---------+---------+---------+---------+---------+
   Setup mode     |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
------------------+---------+---------+---------+---------+---------+---------+
        WNS (ns):| 13.831  | 14.479  | 13.831  |   N/A   |   N/A   |   N/A   |
        TNS (ns):|  0.000  |  0.000  |  0.000  |   N/A   |   N/A   |   N/A   |
  Violating Paths:|    0    |    0    |    0    |   N/A   |   N/A   |   N/A   |
       All Paths:|   534   |   512   |   422   |   N/A   |   N/A   |   N/A   |
------------------+---------+---------+---------+---------+---------+---------+

--------------+-------------------------+--------------+
              |          Real           |     Total    |
  DRVs        +--------------+----------+--------------|
              |Nr nets(terms)| Worst Vio |Nr nets(terms)|
--------------+--------------+----------+--------------+
 max_cap      |    2 (2)     |  -0.083  |    2 (2)     |
 max_tran     |   2 (122)    |  -0.255  |   2 (122)    |
 max_fanout   |    0 (0)     |     0    |    0 (0)     |
--------------+--------------+----------+--------------+

sity: 66.886%
------------------------------------------------------
orted timing to dir timingReports
al CPU time: 1.24 sec
al Real time: 1.0 sec
al Memory Usage: 307.417969 Mbytes
ounter 1>
```



Of course, to make even the **worst violation (worst vio)** positive, we could perform **optimization** multiple times—but this step was skipped.

Another important output of the placement design is the **RC Extraction**, which generates a **.spef** file. This file contains the **parasitic characteristics** of the circuit. We extracted this file from the menu: **Timing → Extract RC**, and it has been **attached**. Other files such as the **GDS file** and the **netlist** were also extracted and attached.

Final Step: We evaluate, through a **trial-and-error approach**, the **smallest possible Die size** suitable for the design. In this stage, various sizes were tested from the beginning, and the best approximate **Die size** for this circuit was determined to be **around 390×390**.