VLSI DESIGN AUTOMATION Project (EC704)

Report

On

PLACEMENT & ROUTING

Submitted by

K. Srikar Siddarth[1] (181EC218)	Sameer S Durgoji[2] (181EC240)

VI SEM B.Tech (ECE)

Under the guidance of

M. S. Bhat
Professor (HAG), Dean (Faculty Welfare)
Dept of ECE, NITK Surathkal

in partial fulfillment for the award of the degree

Bachelor of Technology

in

Electronics and Communication Engineering

At

Department of Electronics and Communication Engineering

National Institute of Technology Karnataka, Surathkal.

JUNE 2021

[1] srikarsiddharth@gmail.com
[2] sameerdurgoji@gmail.com

INDEX

INTRODUCTION

The VLSI physical design converts the circuit description into a geometrical description. The input to this design process will be a netlist and the output will be the layout of the circuit. This geometric description is used to manufacture chips. The physical design consists of the following steps:

1. Partitioning

   Partitioning is decomposition of a complex system into smaller subsystems such that the interconnections among the subsystems is minimized.

2. Floorplanning

   The objective of floorplanning is to find the locations of all the modules as well as their orientations.

3. Placement

   The goal of placement is to find a minimum area arrangement for the blocks that allows completion of interconnections between the blocks, while meeting the performance constraints.

4. Routing

   The objective of the routing phase is to complete the interconnections between blocks according to the specified netlist.

For this project, ISCAS85 Benchmark circuits, s27 and s298 were considered.

DEMO

Steps to run the program:

1. Extract the zip file to a folder. From the folder, find the following files:
   a. placement.py
   b. routing.py
   c. Astar.py

d. c27.txt
e. s27.txt
f. s298.txt

Make sure there is an empty folder named ‘route’ within the unzipped folder. Else please create it.

2. Run the placement.py file using the command

```
python placement.py -c c27.txt -n s298.txt
```

This will do the following:

a. Displays the initial cost of the placement.
b. Plots the cost versus iteration graph.
c. Upon closing the graph, it will display the final cost of the best placement obtained.
d. Saves the finished placement to the file ‘grid.png’
e. Creates a pickle file of the placement as ‘s27_placed.pkl’. This enables the router to use the placement as input.

3. Now run the routing.py file using the same netlist using the command

```
python routing.py -c c27.txt -n s27.txt
```

This will do the following:

a. Generate a pair of routing images at each iteration, one before the routing process is applied and one after the routing is done. These are saved in the ‘route’ folder.

```
phani@phani-desktop:~/siddu/VLSI design automation/project$ python3 placement.py -c c27.txt -n s27.txt
reading config file...
reading netlist file...
generating graph from netlist...
Performing Placement...
initial cost:  38

final cost:  21
Saving Circuit Class Object as a pickle file...
Total Time taken:  2.218183994293213
Placement completed succesfully!!
phani@phani-desktop:~/siddu/VLSI design automation/project$
```

METHODOLOGY

Placement:

The goal of placement is to find a minimum area arrangement for the blocks that allows completion of interconnections between the blocks, while meeting the performance constraints.

The inputs to Placement problem are:

➔ Blocks (standard cells and macros) B1, ... , Bn
➔ Shapes and Pin Positions for each block Bi

- ➔ Nets N1, ... , Nm
- ➔ A netlist of connected gates and nets

The output will be :

- ➔ Coordinates (xi, yi) for block Bi.
- ➔ No overlaps between blocks
- ➔ The total wire length (or the longest wire length) is minimized
- ➔ The area of the resulting block is minimized or given a fixed die
- ➔ Exact location on the chip of each gate

This step was also implemented using Simulated Annealing.

- The move here was to pick a random pair of gates in the grid and exchange their locations.
- The Half Perimeter Wavelength (HPWL) of all the nets was calculated for each iteration.
- The HPWL of a single net can be calculated as shown:

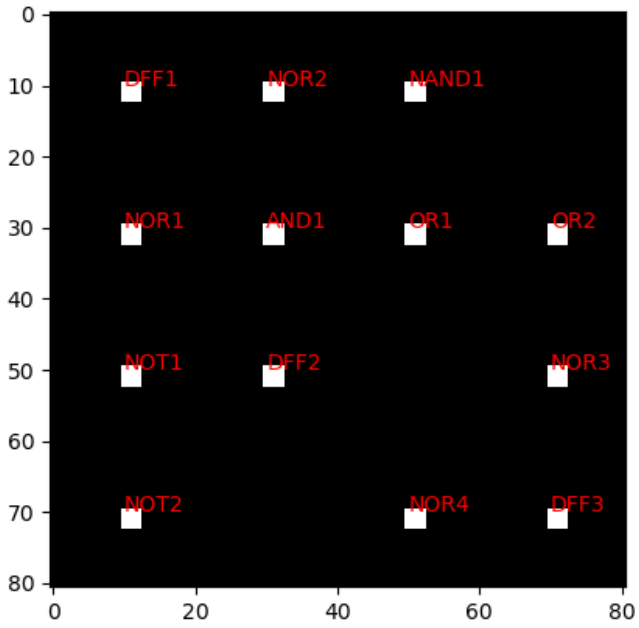
$$\text{HPWL} = [\max\{\text{X coordinates of all gates}) - \min\{\text{X coordinates of all gates}\}] + [\max\{\text{Y coordinates of all gates}) - \min\{\text{Y coordinates of all gates}\}]$$

- The sum of HPWLs of all the nets is calculated as L.
- Then the change in L is calculated as $\Delta L = [\text{new L} - \text{old L}]$

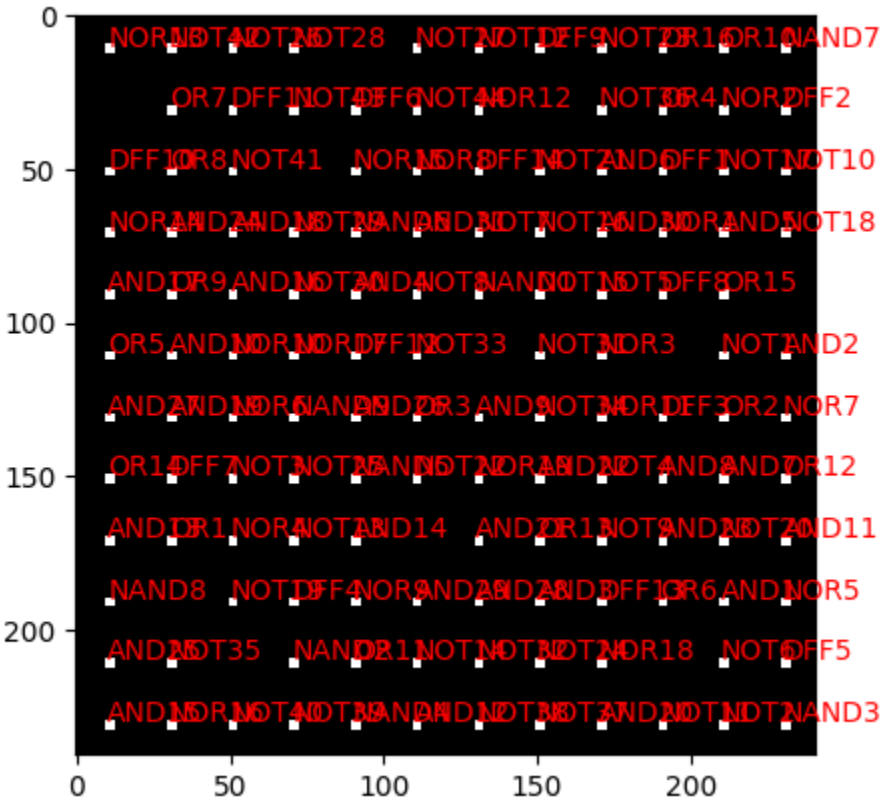
- If $\Delta L < 0$, the new L gets smaller: It will be an improvement and the swap is retained.
- If $\Delta L > 0$, the new L gets bigger: This swap is reversed.

- This is done for many random steps until L stops improving.

Our approach to the problem was to minimize the HPWL cost calculated above. After the simulated annealing is completed, all the details related to the circuit are stored in the pickle file.



The output image after completing the placement for s27.bench.

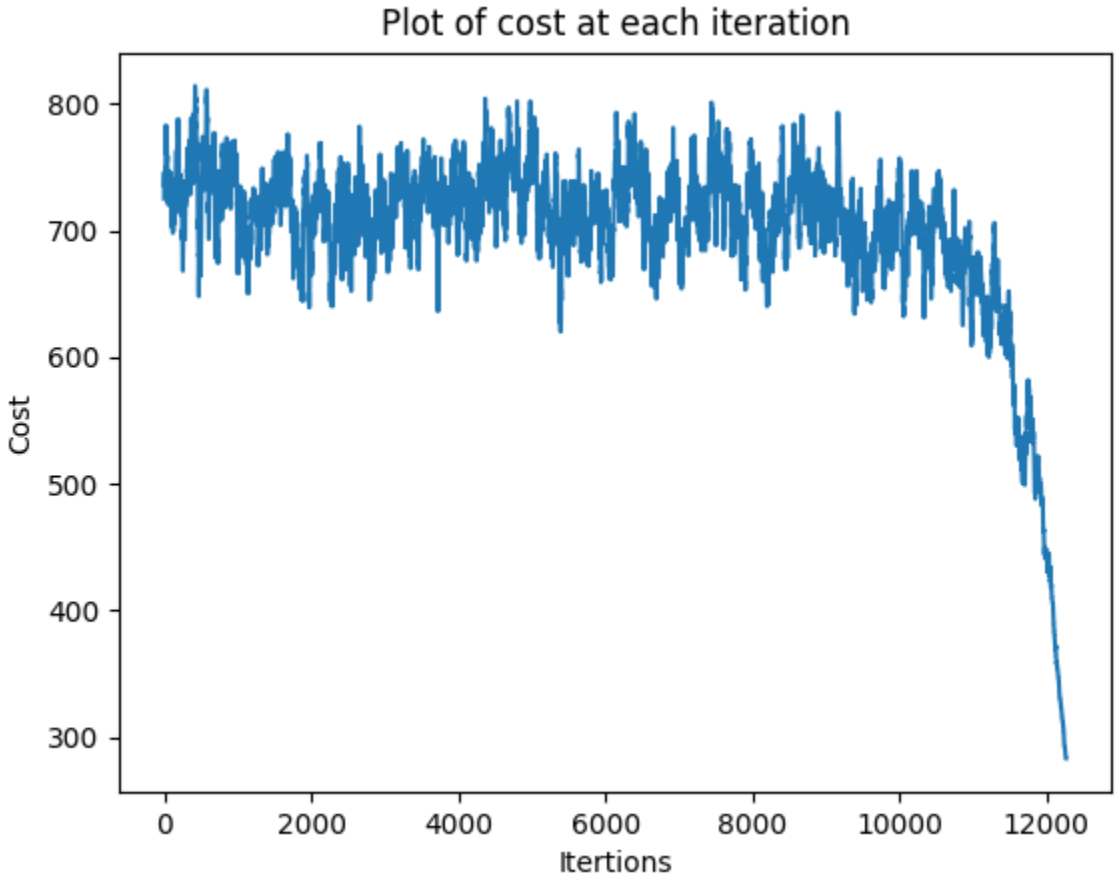


The output image after completing the placement for s298.bench.

```
initial cost:  725

final cost:  283
```

The initial and final costs of placement for s298.bench.



The placement plot of cost at each iteration for s298.bench

Routing:

Now the circuit is ready to be routed. In order to get the minimum possible path, we have made use of the *heap queue* data structure. To calculate the cost of the routing, we have used the *A* (A - star)* algorithm.

A - Star

Input: The grid containing placed blocks, specifically, a 2D numpy array.

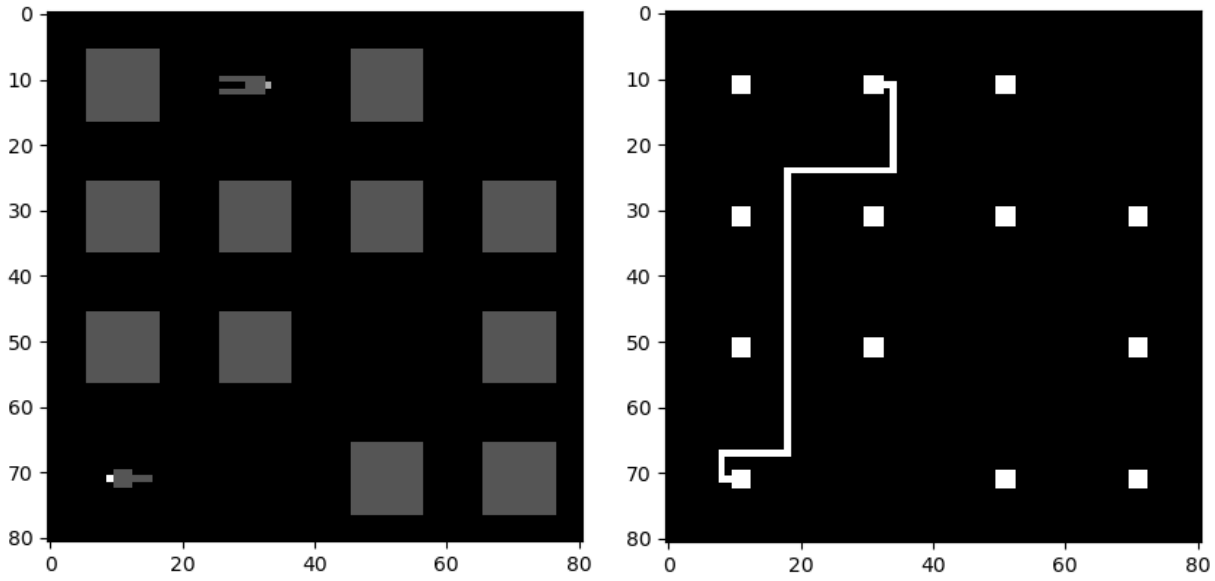
Output: Path (array of points)

Description:

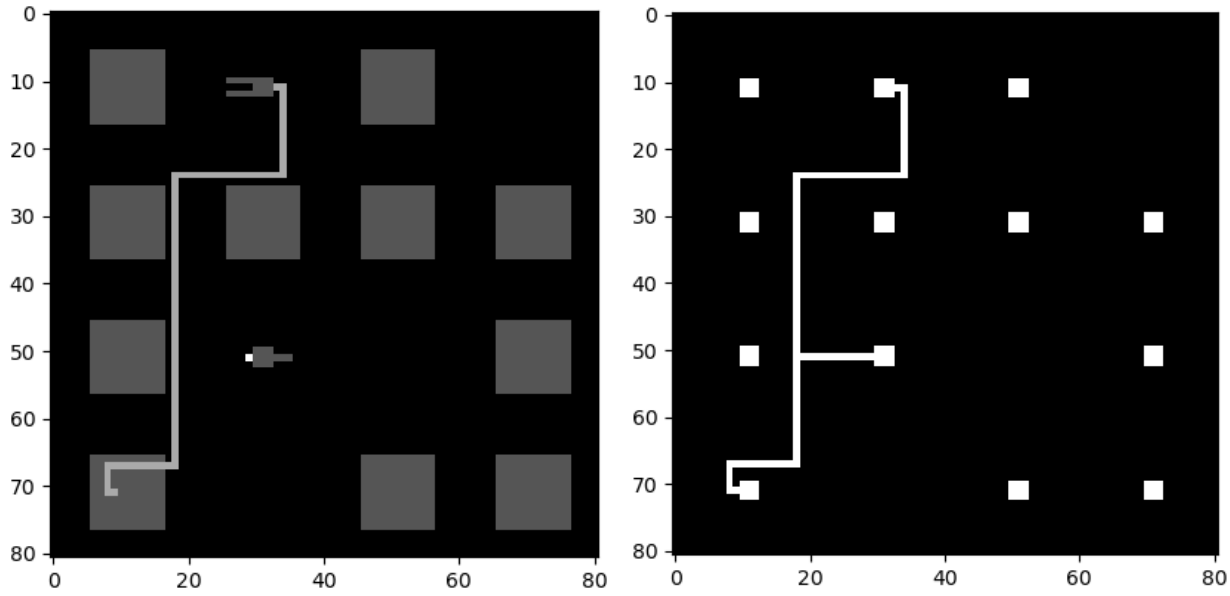
The function considers any cell in the grid as *reachable* if its value is **0**. Else it is marked as not *reachable* if its value is **1**. The starting point of the route can be a 2D-coordinate as well as the set of points. These cells are given a value **2**. The end point can be only a 2D - coordinate with a value of **3**.

Starting the cells marked with **2**, neighbours of each cell are explored and its costs are incremented by 1. If there are bends, then the cost is incremented by 100. Finally the manhattan distance of the current cell to the end cell is calculated and added to the cost. Since we are using a heap queue, the neighbour with the minimum cost is always at the beginning of the queue. So it is easy to get the cell with minimum cost and move ahead to the next cell.

Now before applying the A-star on each edge of the circuit, all the cells except the start and end cell and its pins are given some clearance so that the routing is kept away from these places to avoid congestion (displayed in the left image below in gray color). After the routing is done, these clearances are updated.



The routing process at the first iteration. Circuit before routing(left), circuit after routing(right).



Second iteration of the routing process. Before(left) and after routing(right)

RESULTS

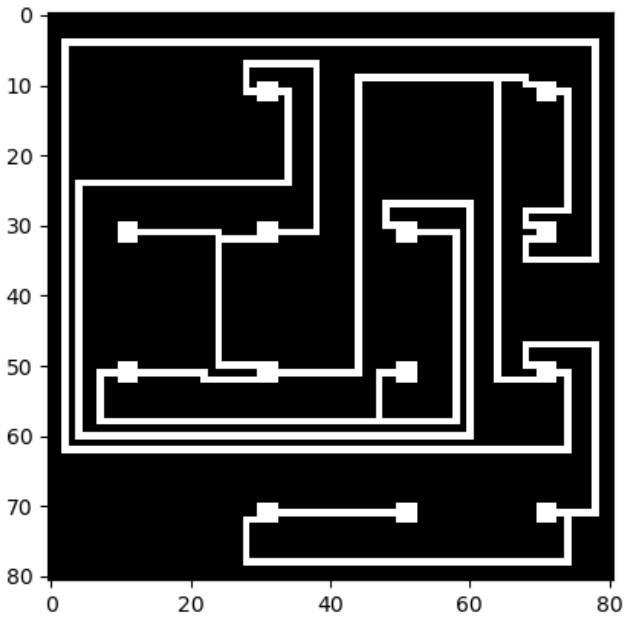
Placement:

Benchmark	Average Initial Cost	Average Final Cost	Time Taken (sec)
s27	37.25	21.75	2.78
s298	752.75	269.5	14.841

The placement algorithm was executed on a desktop in ubuntu OS with intel i5 processor.

Routing:

We are able to route at least 70% of all the paths in a single layer as of now and are working on overcoming the overlapping problems that occurred due to the incorrect ordering of the routes. Below figure describes our most successful routing as of now.



Routing upto 70% of the circuit

CONCLUSION

Various methods of placement and routing were studied and implemented in this project. Various problems encountered while placing and routing were noted and worked upon.

This project is being updated and currently maintained at a *Github* repository. It can be accessed here.