STONY BROOK UNIVERSITY

Department
of
Electrical and Computer Engineering

ESE-556

VLSI Physical and Logic Design Automation

PROJECT- 2

PLACEMENT

Submitted by-
NITIN ASTHANA
SBU ID - 12995559
nitin.asthana@stonybrook.edu

Project-2: Placement

NITIN ASTHANA
SBU ID - 12995559
Graduate student
nitin.asthana@stonybrook.edu
Department of Electrical and Computer Engineering
Stony Brook University, Stony Brook, New York, 11790, US

Problem Statement- To design, implement, and experiment Design, implement, and experiment a placement algorithm with the following characteristics:

- The algorithm implements Fiduccia-Mathesys partitioning-based placement.
- Your code should experiment with all four placement strategies discussed in class. (bisection, quadrature, slice bisection, and cut oriented)
- The algorithm minimizes the total IC area after placement.
- The algorithm maximizes the expected routability of interconnect.
- The algorithm minimizes the expected total interconnect length.
- The algorithm minimizes the expected time delay of the critical nets.

Introduction

Placement is one of the most important step in the vlsi design automation. It is basically a type of design flow in which exact location for various circuit components (i.e. cells) within the IC area is assigned. It is very much important because if the placement assignment is not done properly then the it will not only affect the performance of the integrated chip but bring the situation in which the chip itself might become non-manufacturable as its demand for the wirelength will increase so much that it will be beyond the routing resources. Placement is basically the problem of automatically assigning the correct positions to the predesigned cells on the integrated chips such that there is no overlapping in order to get the optimised objective function. Placement is performed after the design synthesis and is done before the routing. There are basically two types of placement namely standard cell placement and building block placement. The ultimate goal of a placement is to minimise the area and

reduce the size of the interconnect. In a whole we can say the that the placement strategy should be performed in such a way that it should optimised the number of cells so that it meets the circuits performance design.

There are various approaches to the placement strategy which are-

- Partitioning based approach
- Simulated annealing approach
- Analytical approach.

In this report we will mainly discuss the first kind of the approach and try to study the Breuer's Algorithm.

Related Work

TimberWolf algorithm is a standard cell placement and routing algorithm. It arranges the cells in order to minimise the total estimated interconnect cost. It is kind of simulated annealing methodology. The idea of TimberWolf algorithm was inspired from annealing of solids, which leads to a crystal structure. Its packages handle stander cell circuit configurations, in which the standard cells are arranges in horizontal rows and where as many as 11 macro blocks are permitted on chip. Furthermore, the pads are placed around the periphery of the chip.

Breuer's Algorithm- Partitioning Based Placement Algorithm

- He basically proposed the partitioning technique which is used to generate the placement of the cells.
- His idea basically lies on the fact that a given circuit is basically partitioned into two sub-divisions or we can say into sub circuits. We can explain his idea by considering the following points-
    - ✓ At each stage of the partitioning, the total layout area on which the circuits have to be placed is partitioned horizontal and vertical subsections sequentially.
    - ✓ After the partitioning into different half we place the sub circuit into into these sub sections.
    - ✓ The entire process is done recursively and continues until each cell has the unique area in the total layout area.
    - ✓ The criteria by which cut has to be made is same for all the strategy which he proposed i.e. Cutsize is minimised during partitioning.

Below are the four placement strategies which Breuer Algorithm proposes-

1) Quadrature Placement:

In this kind of the partition strategy we first take the entire chip. We then divide the layout into four units using two cut lines, one being horizontal cut line and the other being vertical cut line, both passing through the centre of the layout.

This procedure is then applied to each section of the quarter of the layout received in the previous step recursively. The process is continued until the entire layout is divided into the slots of desired area such that each area have each cell.

Proposed Solution:

- Firstly, we will call our readPlfile function. This function will read the coordinates from the Pl file which is present in each IBM test bench files. From this test file we can place the pads as we have the coordinates of all the pads. Also using the information of the pads coordinates we can find the minimum value of the X-coordinate and maximum value of the X-coordinate. Similarly, we can get the minimum and the maximum value of the Y-coordinate.
- Using this information, we can get the total area of the layout where we must place our cells.
  So, we create calculate the boundaries of the layout.
  Our length = Xmax – Xmin and Breadth = Ymax – Ymin.
  Total Area = Length * Breadth.
- Next, we have to place all the cells in our layout. For this what we will do is we will create a row to cell Map. So, what it does is, it places all the cells row wise and follows the constraints of Ymax so that it does not exceeds the Y limit.
- Also, we will add one more parameter in the map where we stored the cell information. This parameter is -> boolean isPlaced and we will initialise all the cells with the initial value of false.
- Remaining things like reading the Netlist from .net file is same as we did in the project 1 to read the inputs from the files.

Pseudo code for Algorithm:

- Find the minimum and maximum value of the coordinate
- Find the length and breadth and calculate the total area of layout A
- Place all the pads (i/o) pins using the coordinates
- Place all the cells using the row to cell map and place all the cell row wise do the initial placement of all the cells.
- Function quadraturePlacement (Area A, Netlist N) {
- While (for all cells isPlaced == True, total net) {
- Call the Fiduccia Mattheyses algorithm on total net
- Cut the area A into equal area A1 and A2 using the cut line from FM Algo and place N1 and N2
- Call Fiduccia Mattheyses algorithm on N1
- Call Fiduccia Mattheyses algorithm on N2
- Cut Area A1 into two sub-section A1 and A3 and Cut Area A2 into two equal area sub-section A2 and A4 such that the horizontal line is same.
- Check if (area contains only one cell) {
- isPlaced = true;
- Break;
- }
- Call recursive function quadraturePlacement (Area A1, Netlist N1)
- Call recursive function quadraturePlacement (Area A2, Netlist N2)
- Call recursive function quadraturePlacement (Area A3, Netlist N3)
- Call recursive function quadraturePlacement (Area A4, Netlist N4)
- }
- }
- Get the index of all the cells
- Calculate the interconnect length of all the cells and print
- Calculate the area of the layout after placement and print

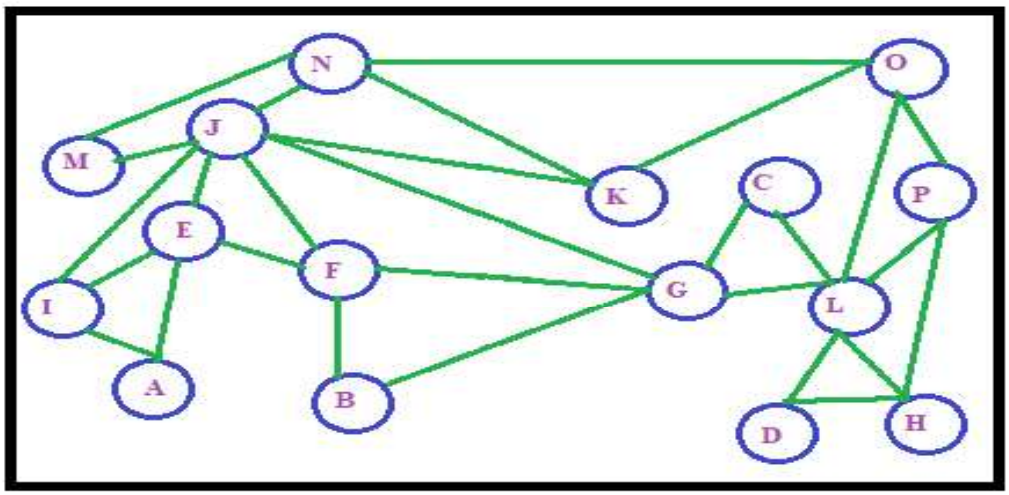
The idea to write the pseudo code for the algorithm has come from the following sample example.

In this example, the layout contains the 16 cells named as A, B,…, P

The circuits for the cells are shown in the first image.

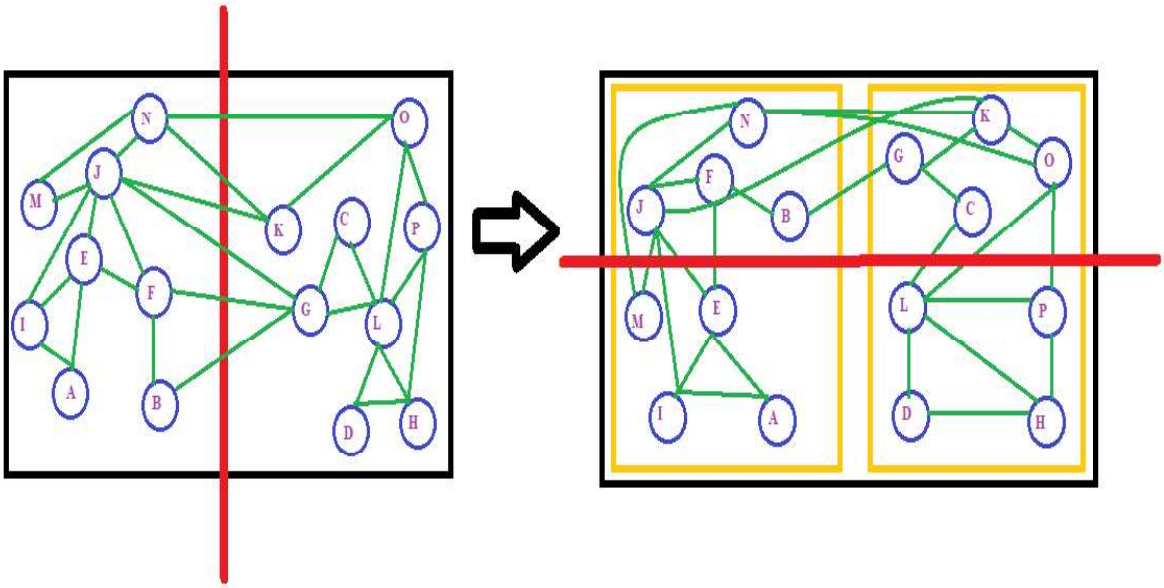
Green color lines depicts the connections.

Red color line depicts the partitioning line.



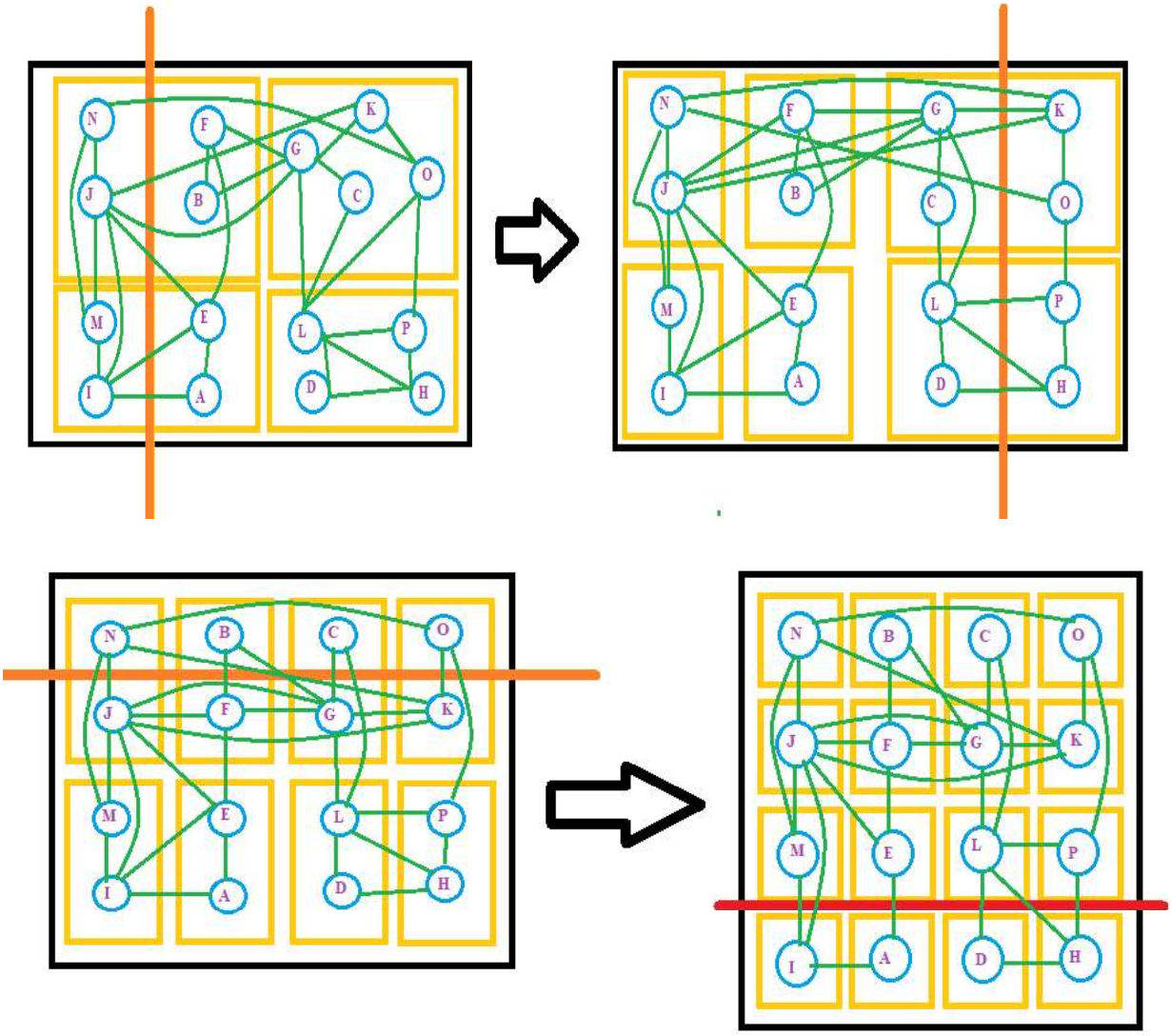
On the above circuit, manually Fiduccia Mattheyses algorithm was implemented, and the first min-cut was found.

After finding the min-cut, the quadrature placement says that the layout has to be cut into four parts by cutting the lines horizontally and vertically from the center.



Therefore, in the first image, the first partition is been done. Then the Fiduccia Mattheyses algorithm was called on both the halves and the second cutline was made and circuit was finally divided into four parts as the demand of the algorithm.

Next, the recursive call has been done to all the parts so that further division can be done.



So finally, after completing all the recursive calls the last image depicts the final placement of all the cells using the quadrature placement strategy.

We can see by comparing it with the first image that overall placement of the cells has been improved and they are arranged in a better manner. Also, since they have come close to each other the over all area of the layout has been decreased. Moreover, as they have come close to each other, the interconnect length has been improved.

2) Cut Oriented Min-Cut Placement:

In this kind of placement strategy, we initially consider the entire chip. We then partition the chip into two blocks. The circuit is also partitioned into the two sub-circuits and the criteria for cutting is that the net cut is minimised. Now the block is then further partitioned by the second cut line and the process is then continued for all the cut lines.

The layout is repeatedly divided recursively using horizontal and vertical cutlines.

Proposed Solution:

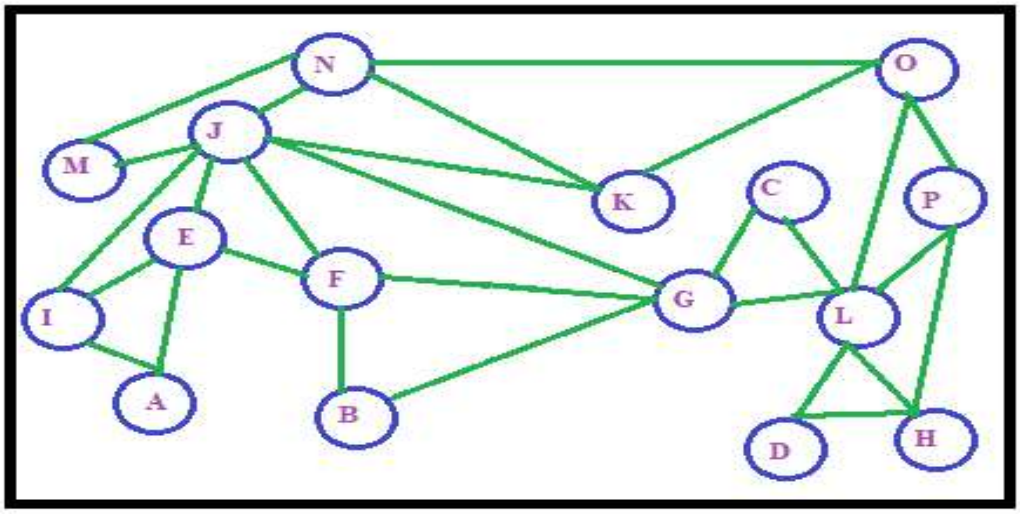
The initial steps of reading of all the input files from the IBM test benchmarks and creating the initial layout and initial placement of the cells are similar to the previous algorithm.

Pseudo code for Algorithm

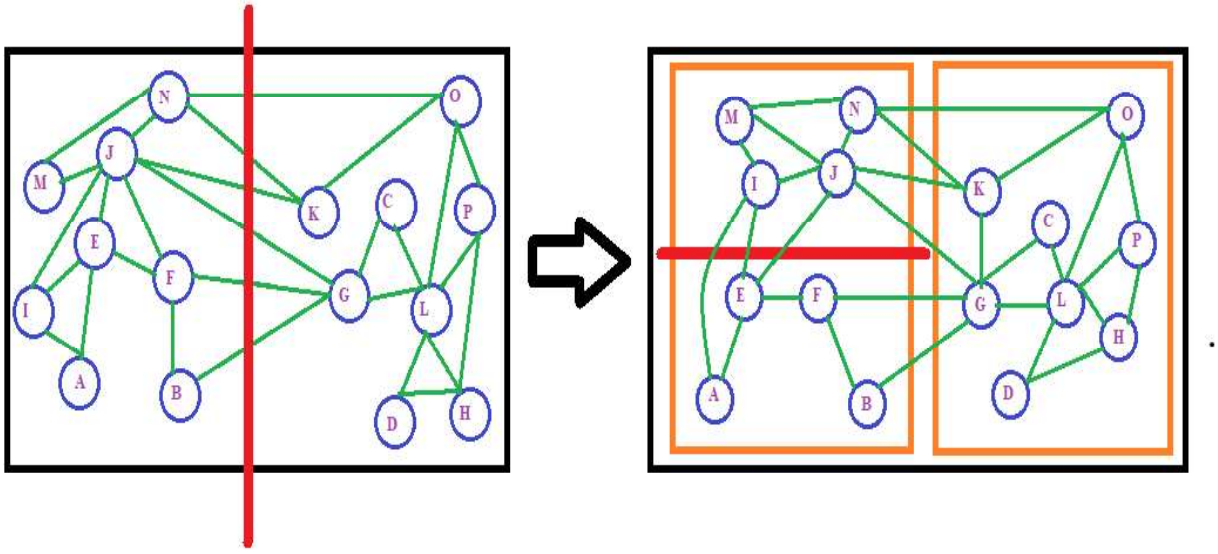
- Find the minimum and maximum value of the coordinate
- Find the length and breath and calculate the total area of layout A
- Place all the pads (i/o) pins using the coordinates
- Place all the cells using the row to cell map and place all the cell row wise do the initial placement of all the cells.
- Function minCut (area A, netlist N) {
- While (for all the cell isPlaced == True, total net) {
- Call the Fiduccia Mattheyses algorithm on total net
  ➔ Total net divided into N1 and N2 netlist with min cut
- Cut the A into A1 and A2 using the cut line got from FM algorithm at that specific coordinate.
- if (number cells in A1 == 1 && number of cells in A2 == 1) {
- cell.isPlaced = True;
- break;
- }
- Call the minCut algorithm recursively on area A1 and net N1;
- Call the minCut algorithm recursively on the area A2 and net N2;
- }
- }
- Get the index of all the cells
- Calculate the interconnect length of all the cells and print
- Calculate the area of the layout after placement and print

Again, the pseudo code for the algorithm is generated using the following example which was performed manually using the FM algorithm for partitioning.

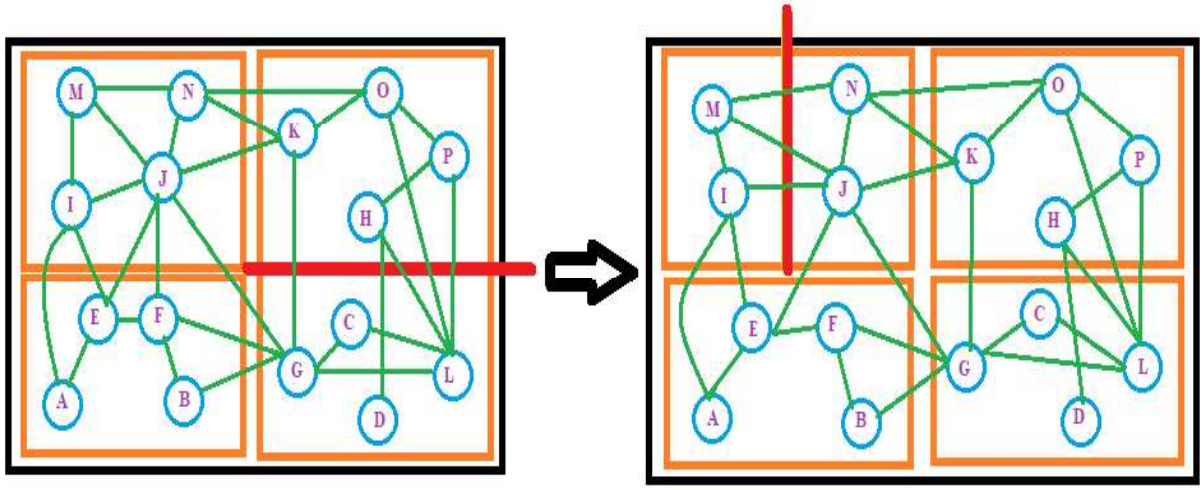
The input circuit is same as the one used in the previous algorithm.

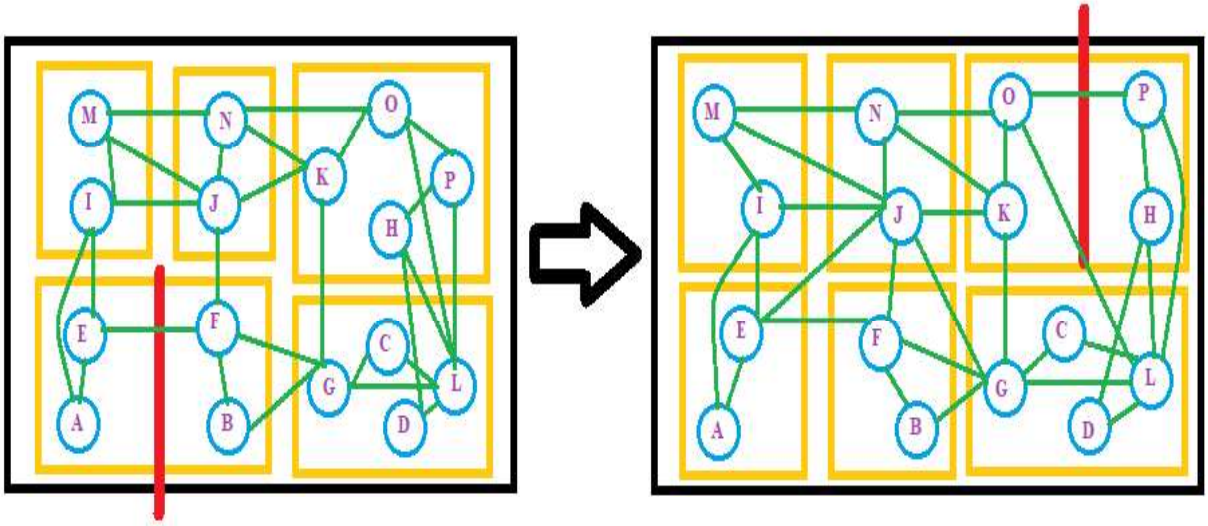


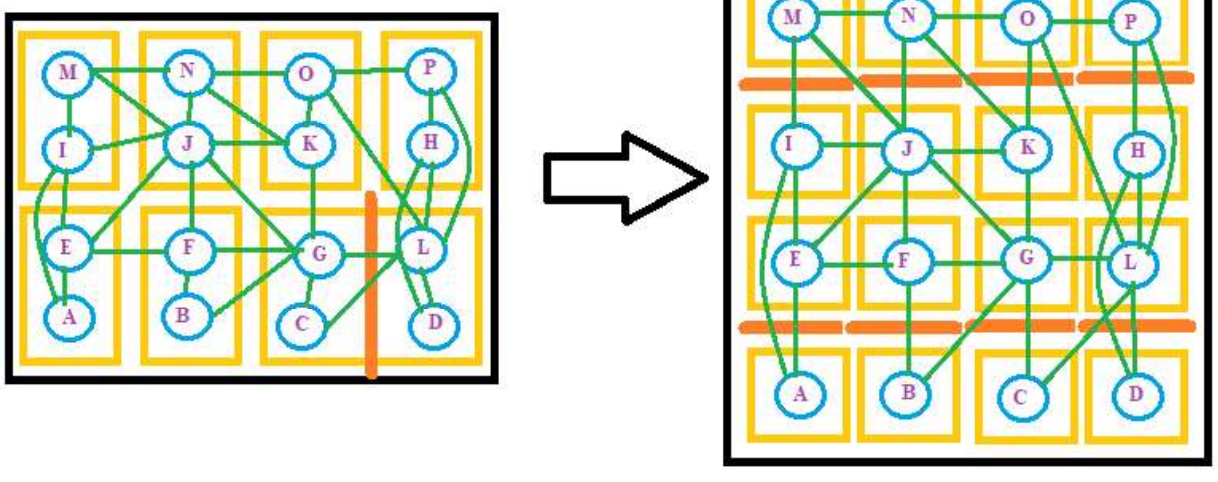
The first step of the algorithm is similar only i.e. on the input circuit the FM algorithm is implemented so as to get the initial min cut such that input layout can be partitioned into the two halves which can be seen in the second image.



Now, the second step varies in this algorithm. Instead of calling the FM on both halves and dividing it with the common line, what we do is we call the FM algorithm in the first half and divide that area into two parts and place the cells. After completing that we then call the FM algorithm to the second half and repeat the same steps further.


M
N
O
K
P
I
J
H
E
F
C
G
L
A
B
D


M
N
O
K
P
I
J
H
E
F
C
G
L
A
B
D


M
N
O
P
I
J
K
H
E
F
G
L
A
B
C
D

After performing the recursive steps, we get the final output in the last image. We can see that final placement is very much better than the initial placement and all the cells have come near to each other and ultimately the total area and the wire length is reduced.

3) Bisection Placement:

In this kind of placement strategy, we repeatedly bisect the layout horizontally until all the cells fits into any of the rows which are generated. We then cut these rows vertically such that the stage is reached where each area generated contains exactly one cell.

Proposed Solution:

The initial steps of reading of all the input files from the IBM test benchmarks and creating the initial layout and initial placement of the cells are similar to the previous algorithm.

Next, in this type of the placement strategy, we do the initial placement randomly. Also, we create two sub-functions which will be called in the main function along with FM algorithm. They are horizontal bisection, which divides the netlist till the point every partition has some cell. In other words, we are saying that all the cells have been placed in some or the other row. The other function which we have created is the vertical bisection. What it does is, it bisects the layout vertically such that each area now generated has only one cell. Following is the pseudo code given.

Pseudo code for Algorithm

- Find the minimum and maximum value of the coordinate
- Find the length and breadth and calculate the total area of layout A.
- Place all the pads (i/o) pins using the coordinates
- Randomly do the initial placement of the cells
- Function horizontal bisection (area A, netlist N) {
- Call FM algorithm and divide the netlist into two parts horizontally;
- Cut layout area into two halves horizontally using min cut;
- Recursively Call Function horizontal bisection (A1, N1);
- Recursively Call Function horizontal bisection (A2, N2);
- }
- Function vertical bisection (area A, netlist N) {
- Call FM algorithm and divide the netlist into two parts vertically
- Cut layout area into two halves horizontally using min cut
- Check if (each area contains cell == 1) {

- isPlaced == True;
- }
- Recursively Call Function vertical bisection (A1, N1);
- Recursively Call Function vertical bisection (A2, N2);
- }
- Function bisection (area A, netlist N) {
- While (for all cells) {
- Function horizontal bisection (area A, netlist N);
- }
- While (isPlaced == true, for all netlist){
- Function vertical bisection (area A, netlist N);
- }
- }
- Get the index of all the cells
- Calculate the interconnect length of all the cells and print
- Calculate the area of the layout after placement and print

4) Slice Bisection Placememt:

This kind of placement strategy is good for cells with high interconnections of periphery. So basically, in this technique what we do is that we take a group of cells and we partition it from the rest of the circuit. We then assign as row to these areas which is called a slice, by horizontal cut lines. To these rows we then do the same vertical partitioning in order to get the area in the layout such that each area contains only one cell. This is strategy is mainly done for circuits which have high degree of inter connections.

Proposed Solution:

The initial steps of reading of all the input files from the IBM test benchmarks and creating the initial layout and initial placement of the cells are similar to the previous algorithm.

Here we have to take some random number to do initial partition. Lets, take 80 percent of the total cells.

Pseudo code for Algorithm

- Find the minimum and maximum value of the coordinate
- Find the length and breadth and calculate the total area of layout A.
- Place all the pads (i/o) pins using the coordinates
- Do the initial placement of the cells
- Function horizontalDivide (Area A, Netlist N) {
- take 80 percent of total cell and call FM algorithm on it.

- Horizontally divide the area into two at that coordinate
- Recursively call the sliceBisection until all cells arranges in rows
- Take the 20 percent cells now do the same to them
- }
- Function verticallDivide (Area A, Netlist N) {
- Call FM algorithm and divide the netlist into two parts vertically
- Cut layout area into two halves horizontally using min cut
- Check if (each area contains cell == 1) {
- isPlaced == True;
- }
- Function sliceBisection (Area A, Netlist N) {
- While (for all cells) {
- Function horizontalDivide (area A, netlist N);
- }
- While (isPlaced == true, for all netlist){
- Function verticallDivide (area A, netlist N);
- }
- }
- Get the index of all the cells
- Calculate the interconnect length of all the cells and print
- Calculate the area of the layout after placement and print

Implementation Issues

The pseudo codes discussed above for the various partitioning based placement strategies could not be implemented successfully. There were many challenges faced in order to code the pseudo codes. I was able to code few functions which are discussed in the pseudo codes and are attached in the appendix. But in all after running everything together, I was not able to get the outputs. I was able to code how the read the input test files properly and few other functions like create initial partition, layout boundaries etc.

One of the issues was there were many things in the input files which needs to be taken care of while coding. I tried to use many data structures but when I tried to run in placement strategy, they were giving errors. I am not sure were the errors were there mainly because of this or there might some there issues as well.

There partitioning based placement algorithm required a lot of hash maps which store the information's like netlist info, cell info, pin number info, coordinate info, rows info etc. I tried to code them but iterating all these maps was a little difficult task to handle. And keep an eye on the working of each of these maps alone was also difficult.

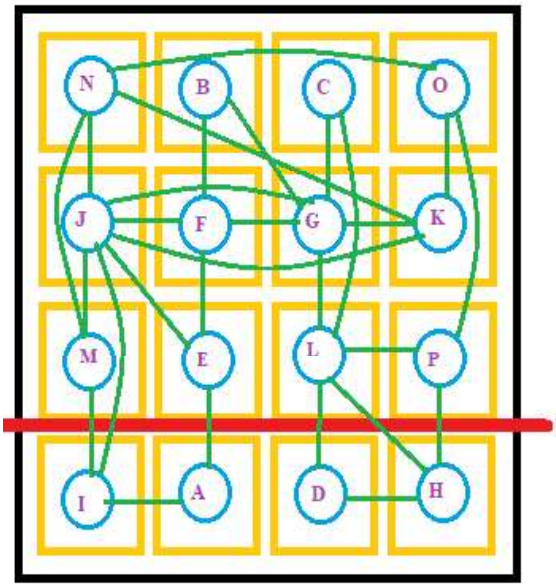
Also, there were many things which I was required to assume while coding. There might be the possibility that these assumptions went wrong.

Next, I tried to change the FM algorithm code which I made in the project 1, so that it can successfully run in this project, but I guess more changes were required. There can also be one more reason that my FM algorithm code was not properly optimised last time. So, calling that entire program as a function might have created a running time error.

In a nutshell a deeper understanding of the topic and selection of the data structures to store the values and handling with malloc etc is required in order to code the pseudo codes properly.

Result

Since I could not implement the placement strategies properly, I can only give some reviews on the results obtained from the manual implementation of the of the placement strategy of the quadrature and min-cut placement technique.



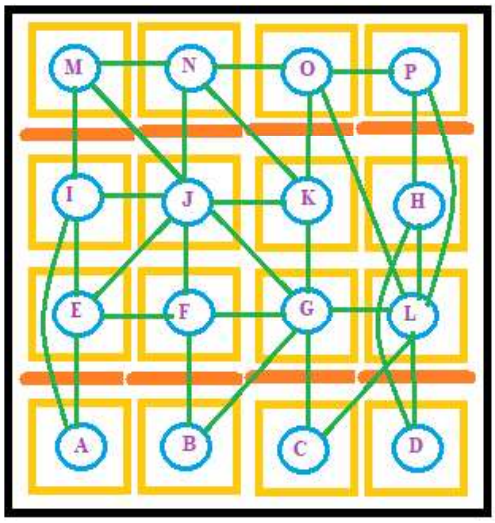


Image 1- Result after the implementation of the quadrature placement.

Image 2- Result after the implementation of the min cut placement.

Above images are the results placement obtained from quadrature and min-cut placement technique. We can see that the quadrature min-cut placement has the better result compared from the quadrature placement strategy as there are lots of wiring in the center of the quadrature one than in comparison to the min-cut one. The placement is done such that each cell is placed so that least wire length to be used for connecting them and out the two we are more successful

in doing that in min-cut placement. In min cut placement there are only three pairs of cells which are connected alternatively whereas quadrature one has six such pairs. So, it is expected that the total IC area after placement is minimised, total interconnect length is also minimised and the routability of the interconnect is maximised. If the pseudo code generated using these observations are correct and if they can be be implemented properly, we can expect that my algorithm will also be able to generate expected output for IBM test files.

Therefore, we can say in this specific example min-cut placement technique is better than the quadrature one. But I cannot generalise the result because this algorithm is not implemented in the IBM test benchmarks. The generalised result might be different if we run all the test cases.

Conclusion

The placement strategies could not be implemented successfully. There is much improvement required in the implementation of the functions of the placement techniques. Much deeper understanding on the topic is required for coding. Also, the FM algorithm needs to be changed more and much more correlation of it with the placement strategy is required. By manual implementation we can say that min-cut placement strategy is better than the quadrature placement strategy. This report was my theoretical understanding of the topic and my thought process in order to make the pseudo codes. To conclude, we can say much more improvement and optimisation is required in order to code the placement strategies properly and compare using IBM test files.

Bibliography

[1] A CLASS OF MIN-CUT PLACEMENT ALGORITHMS (Accessed March 1st, 2020) < http://limsk.ece.gatech.edu/book/papers/breuer.pdf>.

[2] VLSI Cell Placement Techniques by K. SHAHOOKAR AND P. MAZUMDER (Accessed March 1st, 2020) <https://web.eecs.umich.edu/~mazum/PAPERS-MAZUM/cellplacement.pdf>.

[3] Placement by IIT Kharagpur, India (Accessed March 2nd, 2020) <http://www.facweb.iitkgp.ac.in/~isg/CAD/SLIDES/09-placement.pdf>

[4] Fiduccia, C. M. and Mattheyses, R. M. (1982). "A linear-time heuristic for improving network partitions."19th Design Automation Conference, 175–181 (June).

[5] Kernighan Lin algorithm (Accessed on March 1st, 2020) <https://en.wikipedia.org/wiki/KernighanE28093Linalgorithm>.

[6] Physical design (electronics) ((accessed March 1st, 2020) b) <https://en.wikipedia.org/wiki/Physicaldesign(electronics)Partitioning>.

[7] Placement (electronic design automation) (Accessed on March 3rd, 2020) <https://en.wikipedia.org/wiki/Placement_(electronic_design_automation)>.

[8] Min-Cut Floorplacement by Jarrod A. Roy, Saurabh N. Adya, David A. Papa and Igor L. Markov (Accessed on March 4th, 2020) <http://vlsicad.eecs.umich.edu/BK/parquet/TCAD2005_PartPlaceFloor.pdf>.

[9] Placement (Accessed on March 4rd, 2020) <http://users.eecs.northwestern.edu/~haizhou/357/lec4.pdf>.