Implementation

Of

Fiduccia-Mattheyses Partitioning Algorithm

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, USA

***Problem statement*— To implement and experiment the Fiduccia Mattheyses partitioning algorithm Implemented for gate-level designs. The goal of your partitioning project is to minimize the cut set size, while meeting given area constraints fixed for the partitions.**

I. Introduction

Physical design is one of the most important stage in the designing procedure which is there after the circuit design. And in this, partitioning is of the most important part of the process. In VLSI design, partitioning is the process in which the given chip is further divided into the smaller blocks according to the need of the design. In this, the different functional block is placed such a way that they are separated and placed in order to make the routing easier. Partitioning has a significant use in the speed of the designed circuit. Partitioning can be implemented on very large and highly complex circuits. According to Morre's law the number of transistors in a dense integrated circuit approximately doubles every two year and hence the size of VLSI circuit is also is increasing approximate to that rate. Therefore, there is need to improve the implementation of these VLSI circuits.

II. Related Work

A. Kernighan-Lin Algorithm

Kernighan-Lin is an heuristic algorithm i.e. a problem which is designed to solve a known problem quickly when it is difficult to solve using classical approach. The input of the algorithm is a graph G = (V, E). The primary goal of this algorithm is partitioning the vertex set V into two set X and Y and with the minimum cost cut. The algorithm uses an iterative approach and it runs until no further improvement is possible.

B. Fiduccia Mattheyses Algorithm

Fiduucia Mattheyses algorithm is an improvement of the Kernighan-Lin's partitioning algorithm by reducing the time complexity of the algorithm. The algorithm is also an iterative algorithm and it can handle unbalanced partition.

The algorithm introduces the concept of hypergraphs which is an extension of the cut-size from the Kernighan algorithm. In this algorithm, initially all the nodes are stored in a data structure and the gain corresponding to all the nodes are stored in another data structure. Then the nodes are divided into two halves. In every iteration the node is selected which has the maximum gain and it is sent to another partition and it is locked so that it can't be moved in the future. The gain is then updated for all the nodes and process is repeated until all the nodes are locked.

III. Proposed Solution

The design flow of the proposed solution is shown in the following flow chart.

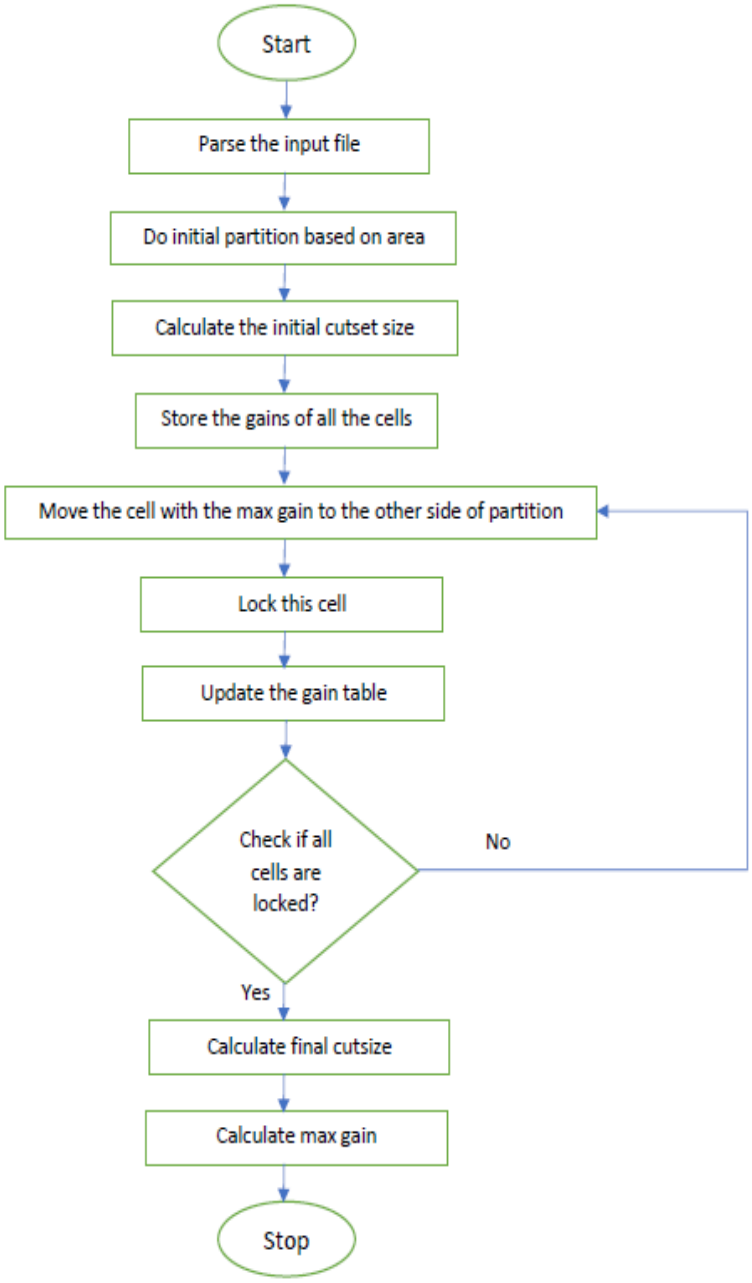

Figure 1. Design flow

IV. Implementation

The implementation of the Fiduccia Mattheyses algorithm was done using C++ on the visual studio. Firstly, the input file is parsed to the code in the main function which were taken from the IBM benchmarks. Then all the nets are added into the hash maps and each key stores another hash map which will store all the nodes connected to that specific net. Next initial partition is done. In this partition, total area of all the cell is calculated and ratio factor is taken into the consideration. If the area is less than half the total it is sent into the 1st partition and if it is greater, then it is sent to the other partition. Now the initial cut size is calculated and printed. Further the bucket list is created which is actually a hash table to store the gain of all the cells. The hash table is having key which stores the gains and the value is vector which contain all the cell corresponding to that gain. The running time of this gain bucket is O (1) as suggested by the Fiducia Mattheyses. Next comes the function which the moves the cell from 1st partition to the other partition. This function iterates the hash table which stores the gain and then take the cell with the maximum gain and moves it to the other side. It does it for the single cell at a time and then locks it. The move cell function does check that whether the cell is fulfilling the area constraints or not. After moving the cell, it locks it so that it doesn't comes into account in future. After doing this, the bucket containing the gains is updated. This process is repeated until no further cell is unlocked. Finally, the cutsize is calculated and final maximum gain is found, and algorithm stops.
Note: I have included the pads in the coding.

V. Implementation Issues

I tried creating my own data structure instead of in-built maps to store the gain, netlist and the cells. But, for creating the similar thing, I was implementing it using the two vectors. The running time was going till O(n) and it was making the code the more complicated. Next for sorting the gains in order to store in the hash table, I tried quick sort and merge sort. But later, found that heap sort is the fastest to find the maximum area. Next tried to do the initial partition randomly. But I was unable to get the sample out itself.

After running the attached code, I was able to run the sample test case which gave the output very frequently. But the first IBM test input gave the output in around 30 mins. Further when I tried to run the next input files, it didn't give the output till two hours of execution and the system was overloaded with the process and could not proceed further. Hence, I feel the current configuration of my laptop is not enough and upgrading it within the short span of time was not feasible for me.

VI. Results

The following results were obtained after running the attached code attached in the appendix. The output for the sample input and the first IBM01 test input is shown below. Although the code didn't give the output for further files, but it was able to execute for test input IBM02 and IBM03 successfully.

Sample input
Initial Cut size: 4
Final Cut size: 2
Maximum gain: 2

IBM01 input
Initial Cut size: 8193
Final Cut size: 6817
Maximum gain: 1376

VII. Conclusion

The Fiduccia Mattheyses algorithm was implemented successfully. The algorithm works perfectly for the smaller input files and gives the expected output. Therefore, it proves the functionality of the algorithm and its correct implementation. However, this code is unable to work for all the IBM test cases. Thus, to conclude, we can say that more improvement and optimisation is required in the code attached.

VIII. Bibliography

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

[2] Kernighan, B. W. and Lin, S. (1970). "An efficient heuristic procedure for partitioning graphs."The Bell System Technical Journal, 49(2), 291–307

[3] (1999).ALENEX â˘A´Z99: Selected Papers from the International Workshop onAlgorithm Engineering and Experimentation, Berlin, Heidelberg. Springer-Verlag.

[4] KernighanLinalgorithm((accessedMarch1, 2020)a),<https://en.wikipedia.org/wiki/KernighanE28093Linalgorithm>.

[5] Physicaldesign(electronics)((accessedMarch1,2020)b),<https://en.wikipedia.org/wiki/Physicaldesign(electronics)Partitioning>.

[6] Cho, W. (2008 (accessed March 1, 2020)).Fiduccia and Mattheyses Algorithm,<https://limsk.ece.gatech.edu/course/ece6133/project/FM.pdf>

STONY BROOK UNIVERSITY

Department
of
Electrical and Computer Engineering

PROJECT- 1
Implementation of Fiduccia-Mattheyses Partitioning Algorithm

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