

CSE 144 - Project Description

It is Year 2013, September. Your illustrious UC San Diego days are long behind and you and your buddy have just been hired at the most promising next generation CAD company, a company whose business plan includes ambitions of unseating the dominant CAD companies and leading the next cycle of VLSI CAD company regeneration this time in Silicon Beach (aka San Diego). You have been given a generous set of stock options with one caveat. You are supposed to implement in not even two months a VLSI layout CAD system (in addition of course to fulfilling all your other obligations in trying to help make sure that the startup survives); your stock options (and your employment contract) become moot unless you exceed expectations on this project. As time is of the essence, you return to UC San Diego to ask for at least a blueprint of the VLSI layout system implementation. Your ex-professor (wizened now with the passage of time) offers you some recommendations, based on what you had studied in his course, CSE 144, in those auspicious spring days of 2008.

While chips have gotten even larger in the intervening time, functionality to be implemented has not failed to keep apace. So even in Year 2013 managers are asking for more functionality than what a chip can accommodate and the prognosis for increased functionality is even more pessimistic. A [partitioning](#) part seems unavoidable; since you had studied both the *Kernighan-Lin* and the *Fiducia-Mattheyses* algorithms in Spring 2008, a partitioning algorithm based on these ideas seems still the safe way to go.

Not only is this algorithm useful for chip partitioning but can also be used in partitioning into rows for a standard-cell design. Standard-cell designs continue to be popular but the design styles have reduced the complexity of placement in Year 2013 and we only need to be concerned about a 2-row design style. Partitioning still seems to be eminently a reasonable algorithm for this 2-row standard cell design style and your professor seems to recommend it, since it is bound to reduce interconnect. Still a [placement](#) within each of the rows needs to be effected. A *simulated annealing* type of algorithm seems to be the way to go; after all your textbook seemed to have almost the whole code written up. Yet a few modifications seem to be needed; the standard cells have unequal size widths and still a metric for evaluating the goodness of the placement needs to be agreed upon.

Finally, to complete the layout system obviously a detailed router is needed. For standard cells, a [channel router](#) is obviously the way to go. After putting all these together, you are satisfied that you now have a clean and doable implementation blueprint and rush back to your company.

Yet in the meantime while you were back on campus, your company has been undergoing severe market pressures and you find that the schedules have been

tightened further. Now you find out that the project is due by **June 2nd, 2008, 1:00PM!** Of course, for a bright young UCSD graduate nothing is impossible. You hop on the nearest time machine, and off you go!

Project overview

You will be finishing the physical design tool for a two-row standard cell design. In the first part of your project, you are required to develop a tool that partitions the circuits to two parts so that each part can be placed in a separate row of the layout. The partition process is subject to some area constraints as the design engineers prefer a somewhat balanced partition which does not generate highly skewed lengths of the two rows. More importantly, in order to facilitate the subsequent placement and routing stages, the number of interconnects that cross the channel between the two rows needs to be minimized. This constitutes the optimization goal of your partition tool.

In the second part of the project, you will be implementing a placement tool which takes the partitioning results as input, and arranges for each component the appropriate position in the row that it belongs to. A good placement needs to reduce the overall interconnect length so as to reduce the circuit delay. The interconnect length between two pins in the placement stage is modeled by the horizontal distance between them. You will be implementing a simulated annealing algorithm to generate the placement that corresponds to minimum interconnect length.

After attaining the placement of the components in the two rows, the final job is to create detailed routing in the channel to connect the components. Your task in the third part of the project is to develop a channel router. Your implementation needs to overcome the major challenge for a channel router, namely, the geometrical constraints that must be fulfilled to avoid generating shorts in interconnects.

Input/Output specifications

The input format is a matrix that models the circuit netlist and area of standard cells as shown in Figure 1. The first line in the file indicates the number of components, N , in the circuit. The following N lines correspond to the connectivity matrix, A , of the circuit, and the last line indicates the sizes of the components in the circuit. The non-zero entries, A_{ij} , in matrix A denote a connection between components i and j in the circuit. For the connection between i and j , A_{ij} indicates the pin number of the component i , and A_{ji} indicates the pin number of the component j .

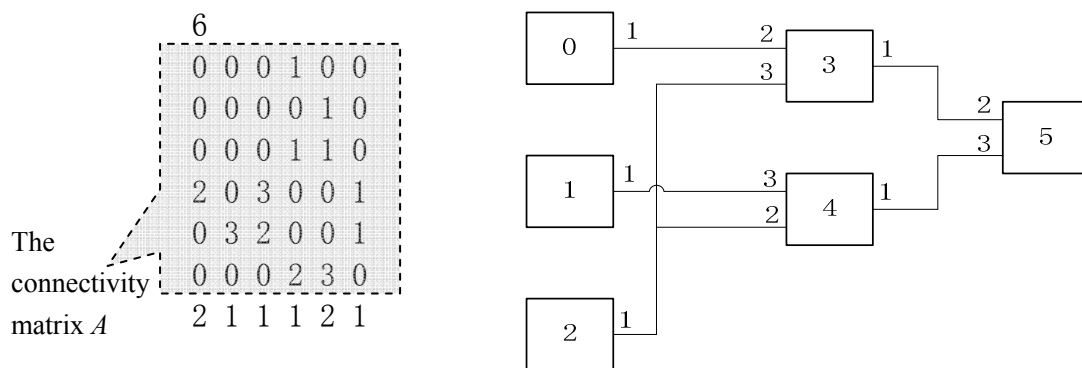


Figure 1 The matrix format of the input

Figure 2 shows the outputs of the partition tool and the placement tool. As you can see, they look quite similar except that the placement results also specify the order in which the components are placed in the layout rows.

Partitioning Results	Placement Results
Partition 0: 0 3 5	Partition 0: 3 5 0
Partition 1: 1 2 4	Partition 1: 2 1 4

Figure 2 The results of partitioning and placement

The output of the router is a list that specifies the characteristics of each net, including the position of pins it connects to, the leftmost and rightmost boundaries of the nets, the routing track it occupies, etc. The following is an example router output. The format will be explained in detail in your Project 3 writeup.

---- Output starts here ----

Partition 0: 3 5 0

Partition 1: 2 1 4

Net 1: (6, 23) [6,23] { 5 } 3

Net 2: (9, 3, 26) [3,26] { } 1

Net 3: (29, 13) [13,29] { } 4

Net 4: (16, 3) [3,16] { 2 } 2

Net 5: (19, 23) [19,23] { } 2

---- Output ends here ----