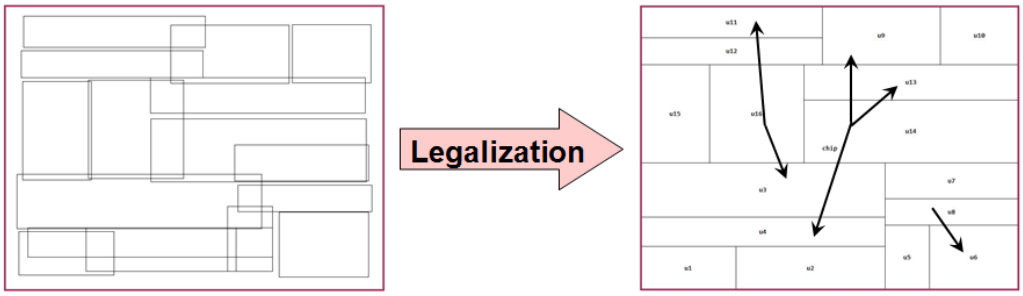
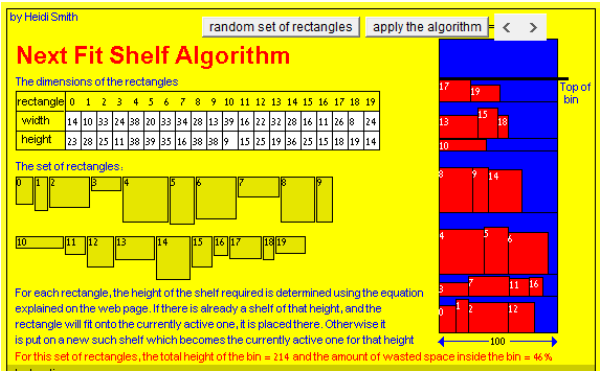
VLSI Floorplan Optimization with Shelf and Level Algorithms

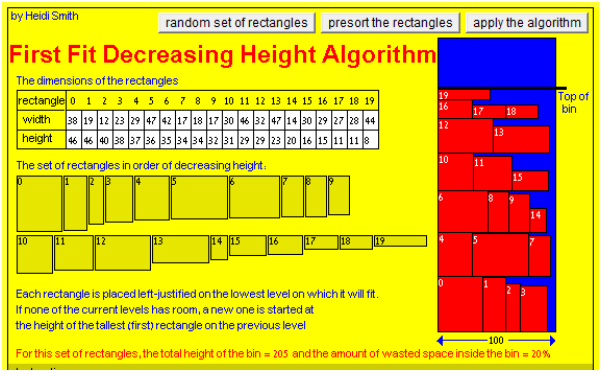
College Mentor: Samy Zafrany

Efficient chip floorplanning is a well-known NP-hard problem with a multitude variety of advanced heuristic algorithms based on simulated annealing techniques and many types of evolutionary algorithms ("genetic algorithms").



However, in many small real life cases, in which the cells have rectangular shape and very similar sizes and aspect ratio, the most fundamental shelf and level algorithms can do a reasonably good job. But it greatly depends on "guessing" a good list of "shelf heights", "next best fit" selection, and optimal chip bounding box ratio. This seems as a reasonable problem for college engineering design project.

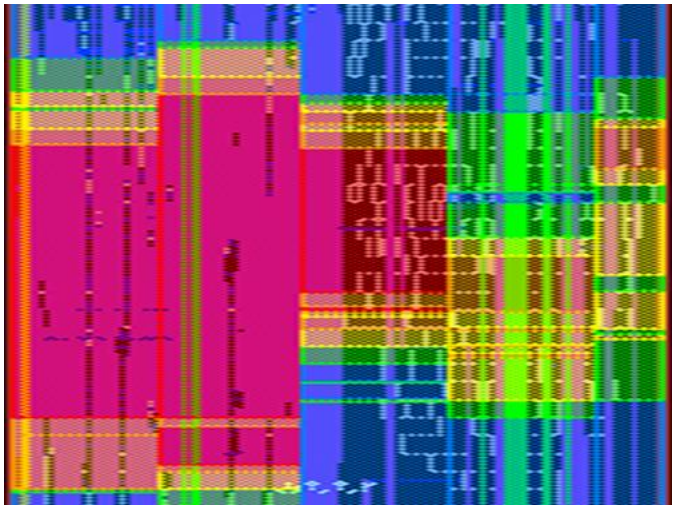




Project aim (student tasks) is to design and implement a system for optimizing the shelf and level algorithms for small scale floorplan problems (no more than 500 cells to place) that has the following properties:

1. System input consists of two files:
    a. Cell file: a list of cell records that we need to place. Consists of cell name, coordinates, and possibly other cell attributes
    b. Netlist file: a list of nets that specify connectivity among cells in the first file

2. The systems should also accept constraint inputs such as:
    a. Chip bounding box size in loose form (such as lower and upper bounds on width and height)
    b. Chip aspect ratio range (lower and upper bound)
    c. Or optionally a combination of area and aspect ratio constraints
3. The system should provide advisory information about *"best shelf heights"* and *"best aspect ratios"* for the chip
4. Optional: the system should minimize the *"average distance"* between connected cells (as specified in the input netlist file). An important part of the project is to define a practical measure of placement quality that takes into account connectivity and timing constraints. If we include this part in the system, then we must also supplement the shelf algorithm with additional operations such as finding "*best permutation*" of shelves that minimizes connectivity distances. Such variations on the traditional shelving process may well exceed the project time budget and it is therefore pending and depends on the breadth of the main target of this project
5. Optional: the system should perform congestion analysis on the resulting floorplans and provide advice on routing quality and feasibility. For this, we may allow an additional input for specifying acceptable congestion levels
6. If needed, we may add a simulated annealing phase to the shelf/level algorithm for achieving better optimized results (in general such hybrid algorithms are helpful for better tuning)



Routing Congestion Map: red regions mark regions in which the amount of routing required exceeds the available area

The system quality will be examined by being applied on real life sets of cells and netlists. Many real examples can be found at the internet, and many other synthetic examples can be created by random

generators. The system outcome will be compared with the expected outcome of known examples and with the outcomes by other industrial tools (if available at the college, academia, or internet).

Outline for Solution Stages

The following list is an estimated path on how to solve the problem and proceed to a successful project completion

1. The first thing is to get acquainted with the basic floor planning concepts, techniques, algorithms, and software tools
   [40 hours]
2. We will be using Python and C++ programming languages and the Tkinter graphical user interface for visualizing VLSI components on top of a TK canvas. The student should invest the needed time to gain intuition, learn and get enough practice in these coding environments before wetting his hands with the real problems
   [20 hours]
3. The student should start with writing simple programs this includes mastering the VLSI GUI drawing tools (such as a basic VLSI canvas), communicating and manipulating VLSI canvas objects
   [20 hours]
4. As a start, the student should implement a simple inefficient algorithm for clustering VLSI units on the basis of overlapping or connectivity relations. The student should be responsible for generating inputs and checkers for checking algorithm validity and code robustness
   [20 hours]
5. Next, the student should be able to implement by himself the most basic level and shelf algorithms for cell placements and VLSI floor planning. This includes generation of appropriate sets of inputs for the algorithms, and an adequate set of checkers, benchmarks, and regression tests for verifying correctness and legalization. Regression tests should cover most of the challenging corner and extreme cases
   [20 hours]
6. The main challenge in this project is to supplement the level and shelf algorithms with connectivity and timing constraints on cells (so connected cells must stay as close to each other). The added complexity in NP-hard, and as such all industrial algorithms are based on heuristic methods, and there's still a lot of room for innovation and improvement. The student will have to come up with one or more ideas and heuristics on how to tackle this problem and then implement it on software
   [100 hours]
7. For that purpose, the student may have to get acquainted with several advanced heurist algorithms such as probabilistic methods, simulated annealing, evolutionary computing (genetic algorithms), and may adopt one of them for his problem - on a smaller scale if they turn to be too powerful for our problem, or the student can come up with a new more appropriate heuristics for the problem. This is yet very open to many directions and we will be more informative once we approach this stage
   [80 hours]
8. The last thing is to run our algorithm on real life cases (know internet industrial test cases) and see how we're doing in several benchmarks such as run time, memory consumption, floor quality, etc. If feasible,

we may try to compare our performance against industrial algorithms if available (although most of them are proprietary and are not publically available)
[50 hours]

The project is yet open for variations and extensions to other type of algorithms and layout quality constraints, and potential students are encouraged to come up with creative ideas, which can make this project suitable for several students (each focuses on a different algorithm/constraint set).

Development durability: One Semester (450 hours)

Resources

1. http://users.cs.cf.ac.uk/C.L.Mumford/Research%20Topics/layout/Outline.html
2. http://www.engr.uconn.edu/~tehrani/teaching/cad/14_floorplanning.pdf
3. http://vlsicad.ucsd.edu/Publications/Journals/j46.pdf
4. http://www.or.uni-bonn.de/~vygen/files/analyt.pdf
5. http://effbot.org/tkinterbook/canvas.htm
6. http://effbot.org/tkinterbook/tkinter-index.htm
7. https://wiki.python.org/moin/TkInter
8. http://effbot.org/tkinterbook/tkinter-index.htm