Report for Programming Assignment #2 Fixed-Outline Floorplanning Using B*-Tree and Simulated Annealing

Student Name: Yi-En Wu
Student ID: B11901188
Course: Physical Design for Nanometer ICs
Instructor: Prof. Yao-Wen Chang

April 25th, 2025

1 Introduction

This assingment asked is to implement a fixed-outline floorplanning algorithm using a B*-tree representation combined with simulated annealing (SA) optimization. The goal is to arrange a set of rectangular blocks within a fixed chip outline while minimizing area and wirelength, and for either prioritizing area or wirelength optimization is defined a parameter, α, by user via command line.

2 Key Algorithms

The floorplan is based on:

- **B*-Tree Representation:** An efficient way to represent and manipulate placements.
- **Packing with Horizontal Contour Data Structure:** Efficiently calculates block coordinates.
- **Simulated Annealing:** Probabilistic optimization to minimize a weighted cost function.

3 Data Structures

3.1 B*-Tree (Tree Class)

- A binary tree.
- Each block is associated with a Node, maintaining pointers to its parent, left child, and right child.
- The Tree manages an array of nodes and a reference to the external Block array.
- Operations include random rotation, delete-and-insert, swap two arbitrary nodes, and contour-based packing.

The definition of B*-tree for packing:

- The left child of a node is placed directly to the right of the parent block, sharing the same bottom contour.
- The right child of a node is placed above the parent block, aligned along the left edge (x-coordinate) of the parent.

3.2 Node Class

The node class is built because of the need of B*-tree. Each node in the B*-tree is a *node* object. Each node will represent a block, this way, the manipulation of each block can be done by operations on node.

- Contains block index, rotation flag, and pointers to parent, left, and right children.

3.3 Contour Structure

- A doubly-linked list of horizontal segments is used to maintain the skyline during packing.
- Supports efficient amortized O(1) insertion and deletion for block placement.

Contour structure is built every time packing happens. When a packing starts, the horizontal contour is incrementally updated during the post-order traversal of the B*-tree.

During packing, the contour segment is used to maintain the current skyline of placed blocks. When a new block is inserted, the algorithm first finds the maximum height over the horizontal interval where the block will be placed to determine its base Y-coordinate. Then, the contour is updated by splitting, removing, or adjusting overlapping segments in that range. A new contour segment representing the newly placed block is inserted with its top height. This ensures that the skyline always reflects the highest occupied positions, enabling efficient O(1) height queries and avoiding block overlaps.

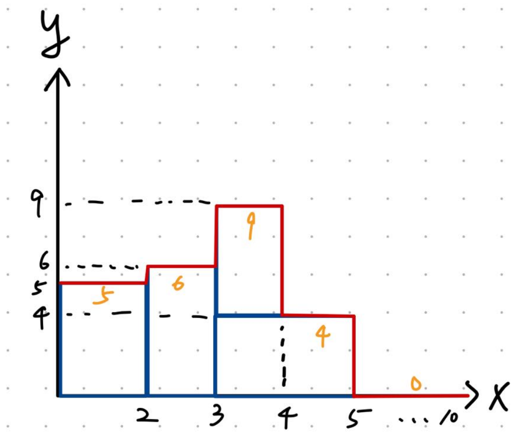
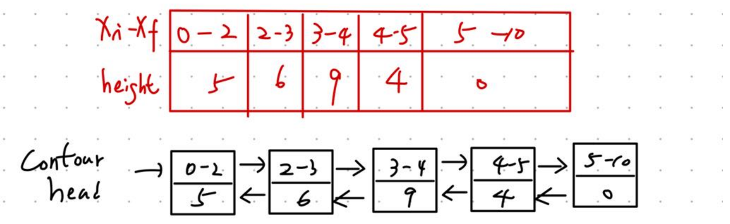


Figure 1: example of a floorplan during packing



Xi-Xf	0 − 2	2-3	3-4	4-5	5 → 10
height	5	6	9	4	0

Figure 2: a doubly-linked list to store the horizontal contour

3.4 Block and Terminal Classes

Block class inherits Terminal class, which is convenience when calculating wirelength. Since the wire-length approximation method in this assignment is by calculating the HPWL, and with both blocks and terminals have stored their x, y coordinates in the terminal object, it makes the calculation fast.

- Block extends Terminal.
- Stores width, height, coordinates, and pointer to associated Node.

3.5 Net Class

- Contains a vector array of *terminals* to store the connected terminals within the net.
- Represents a net connecting multiple terminals(blocks).
- Calculates Half-Perimeter WireLength (HPWL).

4 Important Functions

- **pack()**: Recursive function with contour tracking to determine block placement.
- **swapRandomNodes()**: Handles arbitrary node swaps carefully considering parent-child or ancestor-descendant relationships.
- **deleteAndInsert()**: Deletes a random node and reinserts it into a random feasible position.
- **rotateRandom()**: Randomly toggles a block's rotation.
- **computeCost()**: Combines normalized area, wirelength, aspect ratio deviation, maxX deviation, and maxY deviation.
- **simulatedAnnealing()**: Optimizes the floorplan under decreasing temperature and dynamic adjustment of `_beta` parameter.

5 Floorplan Results

Below are the floorplan results of the five input test cases.
$\alpha = 1$, totally focus on area optimization.

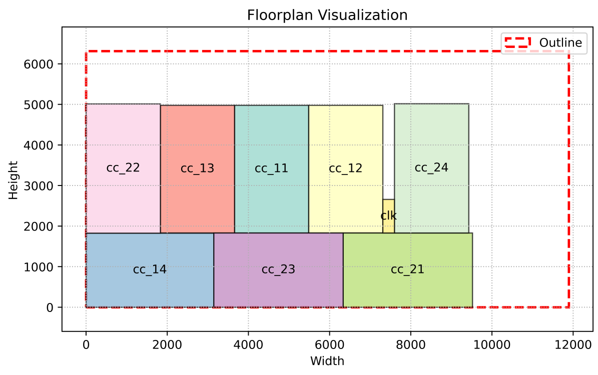


Figure 3: apte

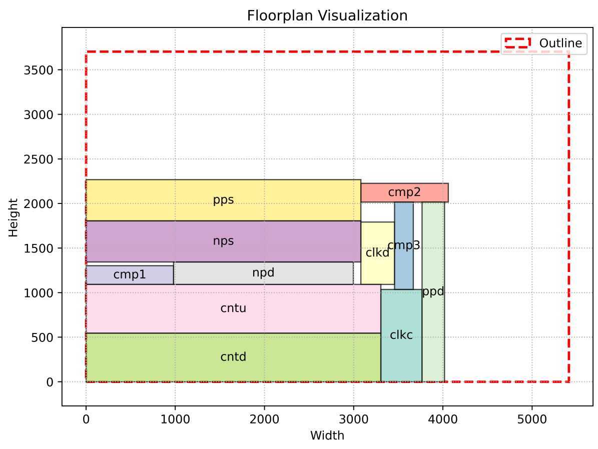


Figure 4: hp

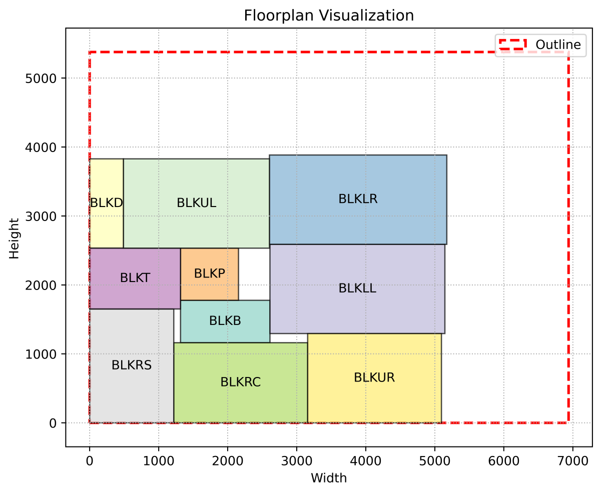


Figure 5: xerox

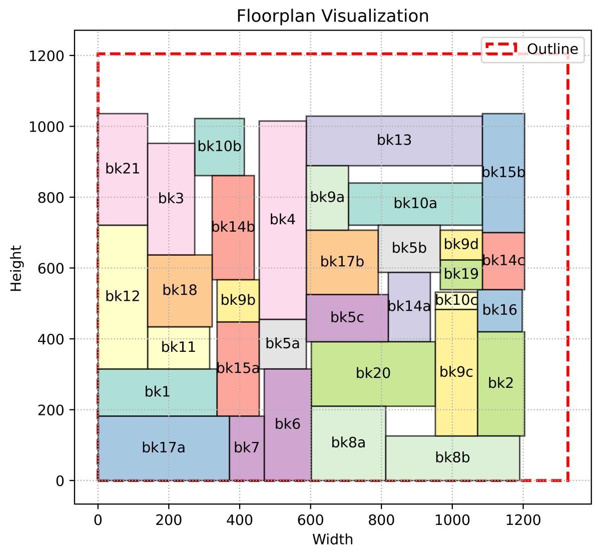


Figure 6: ami33

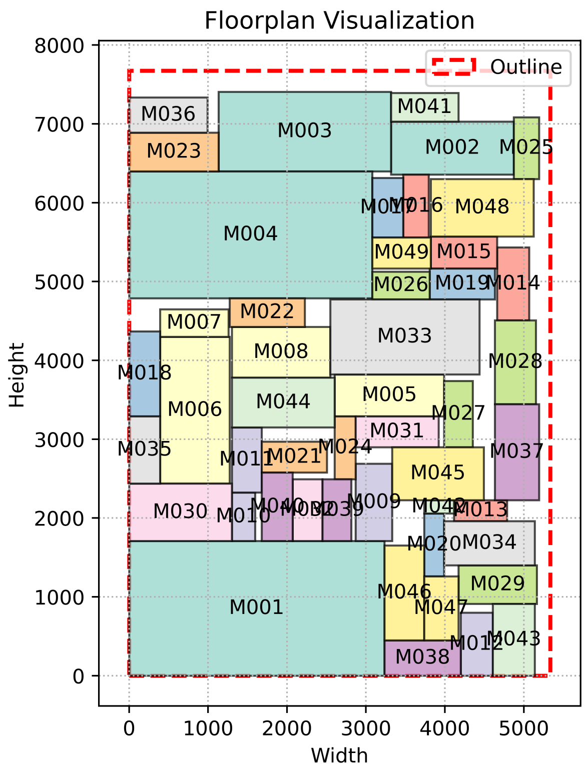


Figure 7: ami49

6 Findings and Observations

- An interesting part is designing a robust swapRandomNodes() that can swap any two nodes while maintaining tree integrity. At first, what I come up is a node swapping function that can only swap nodes that aren't ancestor or descendent to other. However, if the swapping process is limited to this constraint, what I found out is that some nodes(blocks) cannot be reached during the whole simulated annealing process. Thus, after making the node swapping function availible to all arbitrary two nodes, the result became better.

- Allowing adaptive adjustment of `_beta` value during SA improves exploration in early phases and fine-tuning in later phases. What I use is to choose a higher *_beta* initially, this way, I can force the floorplan to meet the require aspect ratio. After iterations, when the cost of aspect ratio term become low enough, i gradually decrease the weight on the aspect ratio term until a pre-set minimum *_beta* value.

- The most challenging problem that I encounter in this assignment is that I cannot make the floorplan to fit into the given fixed outline! What I observe that really help the packing is adding two terms in the cost function, which are, $minX$, and $minY$.
  The definition is as follows:

$$minX = current\_outline\_height \;\; - fixed\_outline\_height$$

$$minY = current\_outline\_width \;\; - fixed\_outline\_width$$

  By defining these two new terms, the SA process will try to fit all the blocks into the required outline to reduce the cost. After adding these two new cost terms, I can finally fit all the blocks into the fixed outline.

- Normalizing area, wirelength, aspect ratio, minX, and mixY in the cost function significantly stabilizes SA behavior. With the normalization and a properly chose weight for each cost term, the SA process can be substantially more effective compared to those without normalization.

- I have change the three perturbation's probability different. For rotating 15%, swapping nodes 50%, and deleting and insert a node 35%. The reason is that swapping nodes tends to have a larger impact on the overall floorplan shape and is crucial for exploring different configurations efficiently, so it is assigned the highest probability. Deleting and reinserting a node also helps restructure the tree but is slightly more disruptive, thus given moderate probability. Rotation, while important for adjusting individual block orientations, affects the overall structure less significantly, so it is applied less frequently to fine-tune local adjustments without destabilizing the global layout too much.
- This is a really interesting assignment; however, I still cannot make the dead space 0, and I currently have no idea how to make it. I think this will be an thoughtful question to think of in the future. What I believe is that maybe the perturbing process can be optimize, such as making the perturbation block dependent, or optimize the probability for different perturbations.