EDA HW4 Report

許育棠 113062556

How to compile and execute

How to Compile

In "HW4/src/", enter the following command:

$ make

How to Run

Usage:

$ ./hw4 <txt file> <out file>

An executable file "hw4" will be generated in "HW4/bin/".

If you want to remove it, please enter the following command:

$ make clean

E.g., in "HW4/bin/", enter the following command:

$ ./hw4 ../testcase/public1.txt ../output/public1.out

Result of HW4_grading.sh

```
checking item             | status
--------------------------|--------
correct tar.gz            | yes
correct file structure    | yes
have README               | yes
have Makefile             | yes
correct make clean        | yes
correct make              | yes

 testcase |         area |    runtime | status
----------|--------------|------------|--------
  public1 |     47448800 |       5.09 | success
  public2 |       655338 |      17.73 | success
  public3 |       659061 |      44.81 | success
+-------------------------------------------------+
|                                                 |
|   Successfully write grades to HW4_grade.csv    |
|                                                 |
+-------------------------------------------------+
```

Implementation Details

1. Symmetry constraint

I handle symmetry constraints by grouping all symmetric modules into a symmetry island, ensuring that nodes of the same symmetry group are placed adjacently. To implement this, I extend the standard B*-tree with a HierNode class that inherits from the base Node. Each symmetry group becomes one HierNode, and inside it I build a local B*-tree that represents only half of the group (right side of a vertical cut). For each symmetric pair, I insert just one of the two modules into this local tree; the other is positioned later by mirroring its partner during the final packing step. This hierarchical B*-tree structure lets me enforce symmetry islands efficiently while still benefiting from the B*-tree's proven floorplanning performance.

2. Initial solution

As mentioned above, each HierNode encapsulates a symmetry island with its own local B*-tree, which we initialize by first stacking all self-symmetric modules in a vertical column, then stacking one representative from each symmetric pair immediately to the right of that column to form a contiguous "half-layout." We treat this ordering as the initial preorder/inorder of the local B*-tree, build it, and then run simulated annealing to compact its area; the final packed module positions define the island's bounding-box width and height. To form the global initial solution, we take every HierNode (and any standalone hard blocks) in some fixed order, designate the first as the root, attach the next two as its left and right children, then continue down the list pairing each subsequent parent with two children in turn. Finally, extract preorder and inorder by traversing the tree.

3. Simulated annealing

There are two situations where SA is used, minimizing the area inside the local B* tree and minimizing the area of global B* tree. The implementation of simulated annealing is the same as HW3, including parameter settings. However, the most difficult part is that we have to clone every node/HierNode to preserve the best B* tree during SA.

4. Perturb

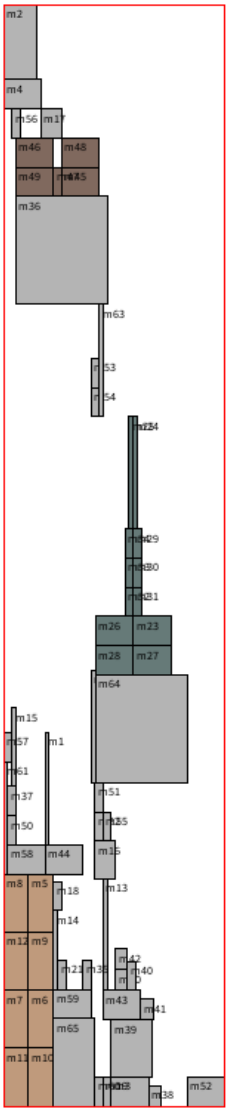
I implement three perturbation methods to modify the B* tree. 1. Rotate a non-HierNode 2. Swap two nodes 3. Reinsert a Node. The first and second methods can be simply applied to preorder and inorder of the tree. However, the third method needs to traverse the whole tree to get the perturbed preorder and inorder of the tree.

Initial Floorplan Method

Answered in previous question.

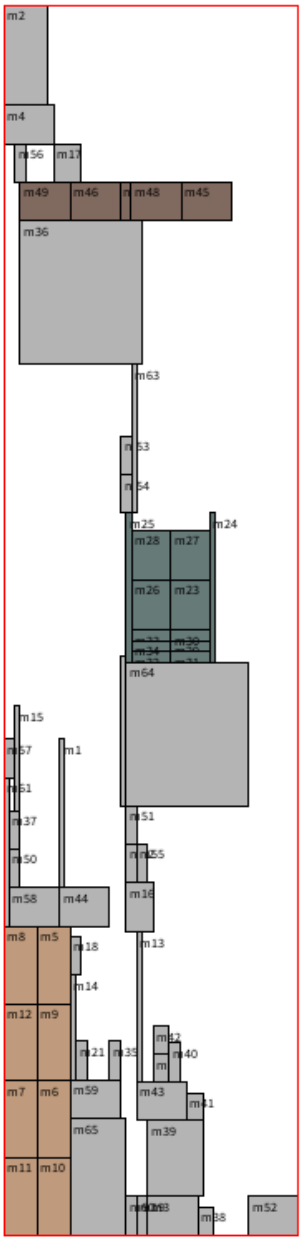
Tricks Used (use public2 as example)

1. No optimization



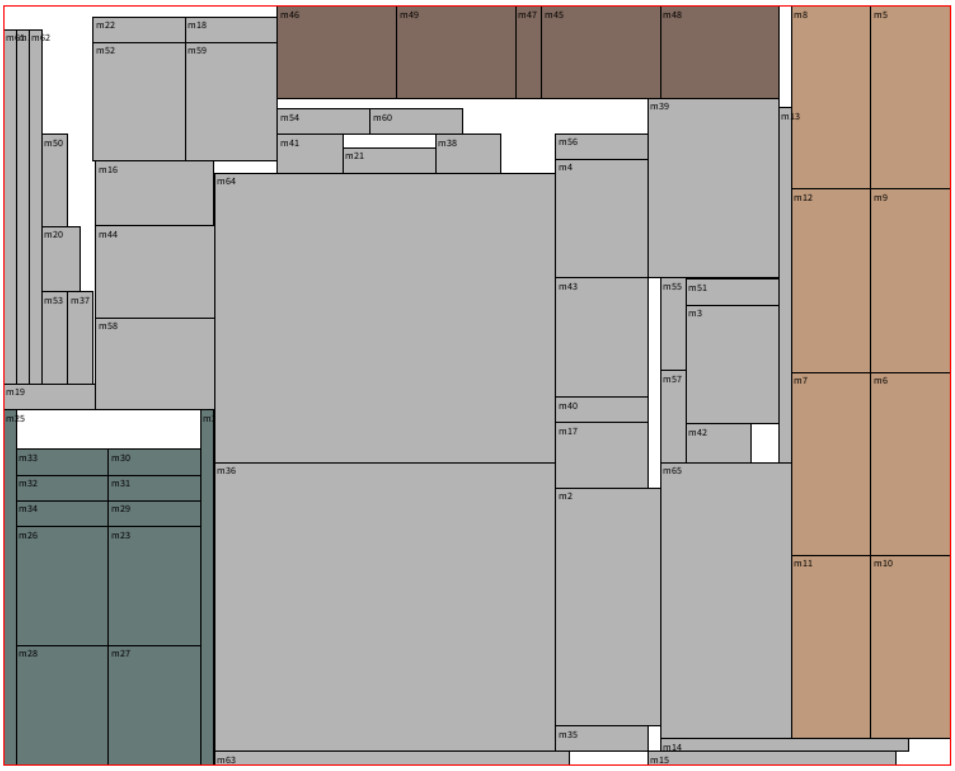
After initial floorplan, as you can see, the dead space is large.

2. Minimizing local B* tree



After using SA to minimize local B* tree, we can see the dead space within symmetry island is minimized. (Take a look at the green symmetry island)

3. Minimizing both local and global B* tree



The global simulated annealing will then minimize the dead space outside of symmetry island.

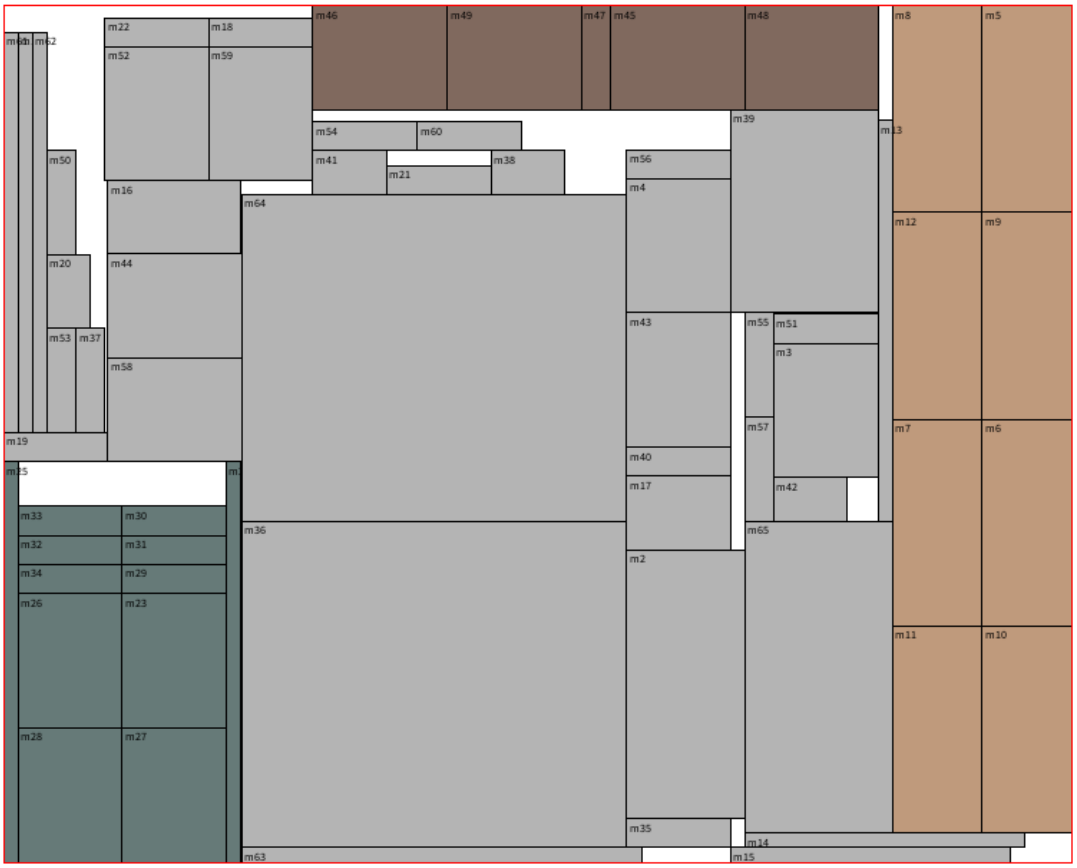
What Have I Learned

Unlike my previous assignments—which were all about re-implementing standard optimization algorithms—this project challenged me to extend the TA-provided B*-tree code into a brand-new data structure. I spent most of my effort designing a hierarchical B*-tree that could naturally handle symmetry constraints. I learned how to solve the symmetry constraint in analog placement which is really different from digital placement. There are still plenty of places to refine my design—for example, I simplified each symmetry island into a single hard block, whereas the original paper uses a more fine-grained treatment of pairs. Even so, working through this assignment let me learn a lot about data-structure design, symmetry handling, and floorplan optimization.

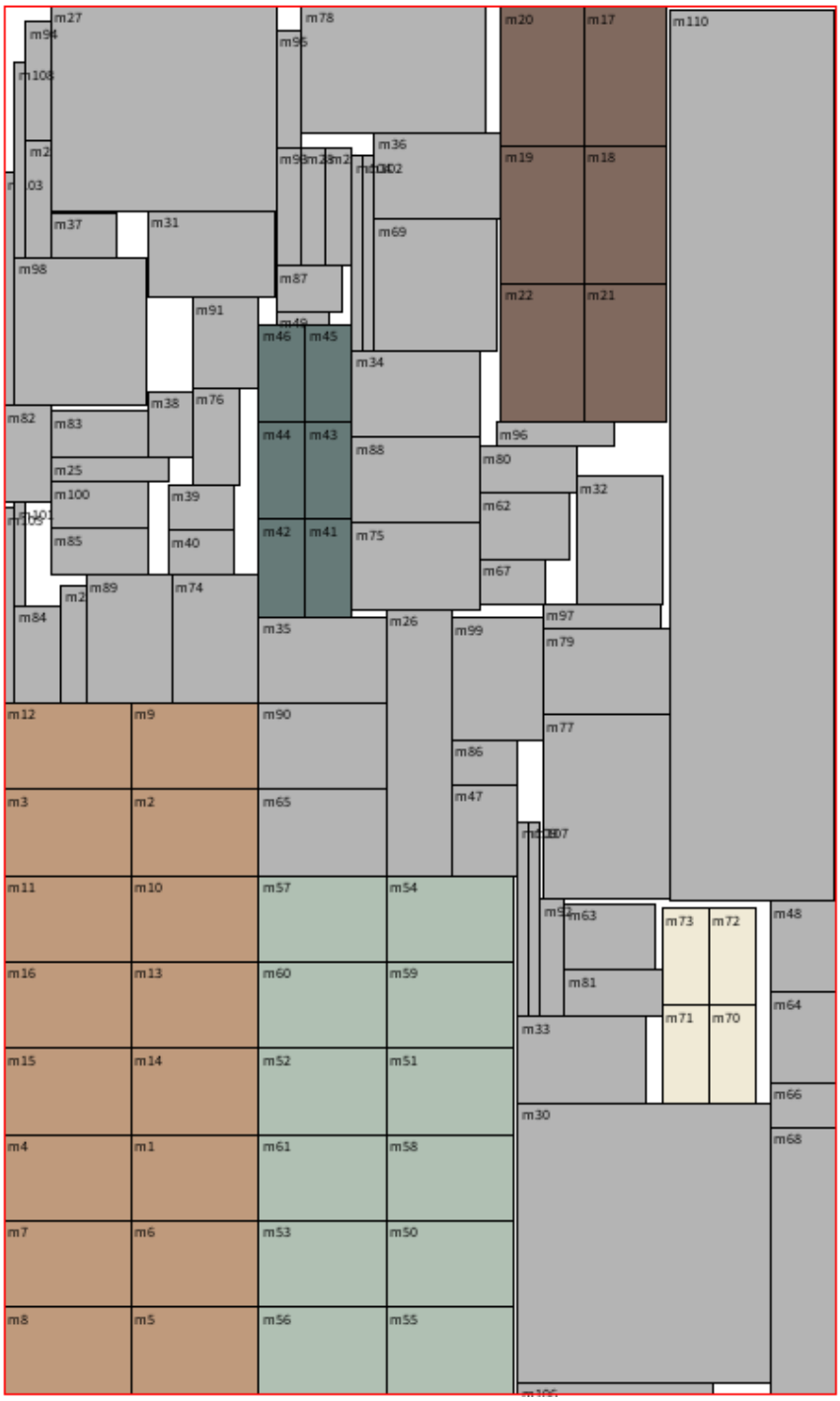
Final Results for Each Testcase

Public1:

Public2:



Public3:


m27
m94
m108
m78
m96
m20
m17
m110
m36
m19
m18
m37
m31
m69
m98
m87
m22
m21
m91
m46
m45
m34
m38
m76
m82
m83
m44
m43
m88
m96
m80
m25
m32
m100
m39
m62
m85
m40
m42
m41
m75
m67
m89
m74
m84
m35
m26
m99
m97
m79
m12
m9
m90
m77
m86
m3
m2
m65
m47
m11
m10
m57
m54
m63
m73
m72
m48
m16
m13
m60
m59
m81
m64
m33
m71
m70
m15
m14
m52
m51
m66
m30
m4
m1
m61
m58
m68
m7
m6
m53
m50
m8
m5
m56
m55