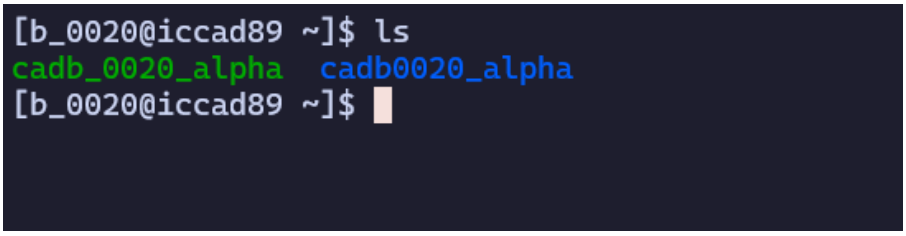
EDA Final Project Report

110511010 楊育陞 110511067 葉哲伍



ICCAD Contest 繳交截圖

1 Introduction

We take the **problem B** of the ICCAD contest as our final project. The problem is "Power and Timing Optimization Using Multibit Flip-Flop". Our method to solve this problem is mainly based on **clustering** and **gain-based greedy algorithm**. The clustering algorithm we use is inspired by the "Effective Mean Shift Algorithm" described in the paper "Graceful Register Clustering by Effective Mean Shift Algorithm for Power and Timing Balancing" [1].

2 Method

2.1 Flowchart

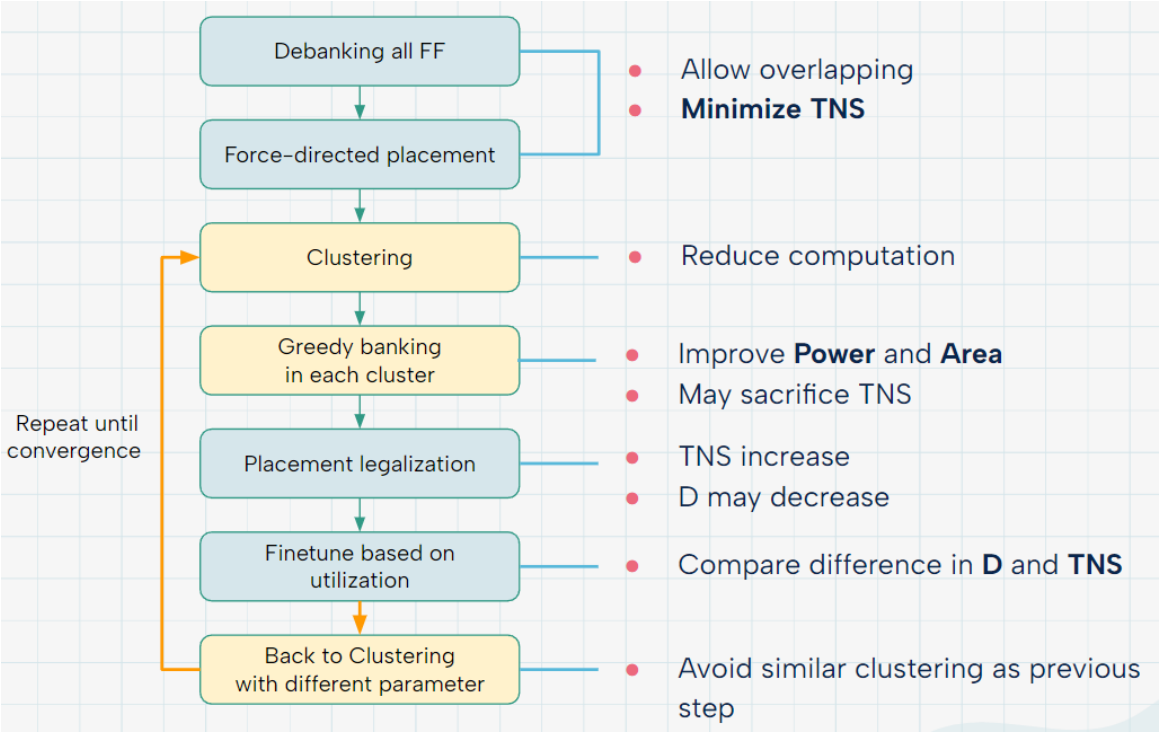


Figure 1: Flowchart of our method

2.2 Division of Labor

- 楊育陞: Debanking, Greedy banking, Finetuning based on utilzation
- 葉哲伍: Force-directed placement, Clustering, Placement legalization

2.3 Detailed Explanation

Our flow of the method is shown in Figure 1. We will explain each step in detail in the following subsections.

2.3.1 Debanking all flip-flops

In the initial given circuit, there are flip-flops of different bit-widths and different types. We first debank all flip-flops to single-bit flip-flops. The type of the single-bit flip-flop, called **base flip-flop** in our method, is chosen to be the single-bit flip-flop with the least cost calculated by the given cost function. The timing part of the cost is calculated by the q pin delay of the flip-flop in this step.

The debanking process is shown in Figure 2. The debanked flip-flops are placed in the same position as the original flip-flops, and the connections are also kept the same. In this step, we allow the cells to overlap with each other.

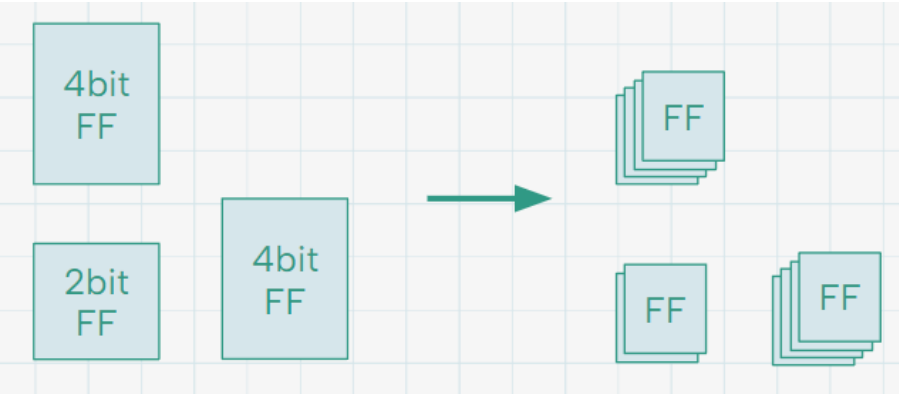


Figure 2: Debanking process

2.3.2 Force-directed placement

After debanking the flip-flops, we perform force-directed placement to determine the placement of the single-bit flip-flops. The force(weight) of each net is determined by the slack of the net. The first debanking step and this step are used to optimize the timing condition of the circuit.

In this problem, the cells are split into two types, combinational cells and flip-flops. The combinational cells are not allowed to move. So we should deal with two conditions when moving a flip-flop:

- **All the connected cells of this flip-flop are combinational cells:** In this case, we can directly move the flip-flop to the timing optimal position.
- **There are flip-flops in the connected cells:** In this case, directly moving the flip-flop may not achieve the best timing condition since other connected flip-flops may move as

well. So we should consider the movement of the connected flip-flops and the movement of the flip-flop itself. We use a iterative greedy method to solve this problem: We move one flip-flop at a time and fix the position of the other flip-flops. We repeat this process until the positions of all flip-flops are fixed or the distances converge.

For this step, we introduce a **"lock" mechanism** to achieve high efficiency. The lock mechanism is used to lock the position of the flip-flops that have been placed in the optimal position or the distances for several iterations have converged. The locked flip-flops will not be moved in the following iterations. Since the condition 1 discussed above will lock the flip-flop in the first iteration, **there is no need to consider which condition the flip-flop belongs to**, highly improving the efficiency of the algorithm.

2.3.3 Clustering

Our clustering algorithm is inspired by the "Effective Mean Shift Algorithm" described in the paper "Graceful Register Clustering by Effective Mean Shift Algorithm for Power and Timing Balancing" [1]. However, we have made some modifications to adapt it to our specific needs. **Instead of directly applying the clustering algorithm for power and timing balancing, we use it to create clusters of flip-flops for our next step, which is greedy banking.** This allows us to optimize the time complexity of the greedy banking process by performing it on smaller clusters rather than on all flip-flops at once. Additionally, we believe that the greedy banking process should only consider nearby flip-flops within each cluster, which further improves its efficiency.

In this step, there are two part of parameters to be tuned:

- **Bandwidth:** The bandwidth of the clustering algorithm. This parameter determines the size of the cluster. A larger bandwidth will result in fewer clusters, while a smaller bandwidth will result in more clusters. Fewer clusters may lead to a better banking result, but it also increases the time complexity of the greedy banking process.
- **Effective Negihbors:** The number of effective neighbors of a flip-flop. This parameter determines the number of flip-flops that are considered as neighbors of a flip-flop when calculating the mean shift. This parameter mainly affects the runtime of the clustering algorithm.

2.3.4 Greedy banking in each cluster

After clustering the flip-flops, we perform the greedy banking algorithm in each cluster. The greedy banking algorithm is as follows:

1. Calculate the gain of all pairs of flip-flops in the cluster. The gain of a pair of flip-flops is defined as **the reduction of the cost function if the two flip-flops are banked together and placed in the optimal position using the force-directed placement algorithm**. The cost function is provided in the problem as follows, here the penalty term D is ignored in this step:

$$\sum_{\forall i \in FF} (\alpha \cdot TNS(i) + \beta \cdot Power(i) + \gamma \cdot Area(i)) + \lambda \cdot D$$

2. Choose the pair of flip-flops with the highest gain and bank them together.
3. Repeat step 1 and 2 until no pair of flip-flops has a positive gain.

This step aims to deal with that the cost function differs in each testcase. But this might lead to a local optimal solution as we only consider choosing the pair with positive gain.

After the greedy banking process, the positions of the flip-flops are updated, and the connections are also updated accordingly. This may lead to non-optimal timing conditions in this state. So we need to **perform the force-directed placement algorithm again** to optimize the timing condition.

2.3.5 Placement legalization

The above steps would lead to illegal placements as they allow the flip-flops to overlap with each other. So we need to perform placement legalization to avoid the overlap of flip-flops. Our placement legalization algorithm is as follows:

Starting from the bottom row, for each row:

1. Remove every FF from the row and sort the FFs based on their x coordinates.
2. Scan the row from left to right and place the FFs if available.
3. If the scan is done and there are remaining FFs, store them as orphans.

Once all rows are finished, find the available sites for orphans by performing a BFS from their original positions.

2.3.6 Finetuning based on utilization

In the cost function provided in the problem, there is a penalty term D that is related to the utilization rate of cells in the bins. All the above steps ignore this term. So we fineture the placement based on the utilization rate to optimize the cost function in this step. Consider one bin at a time, if the utilization rate of the bin is higher than a threshold, we try to move the flip-flops in the bin to other bins to reduce the utilization rate. This would lead to a higher cost as the change of wirelength. **If the cost function is reduced, we accept the move, otherwise we reject the move.** We repeat this process until the cost converges.

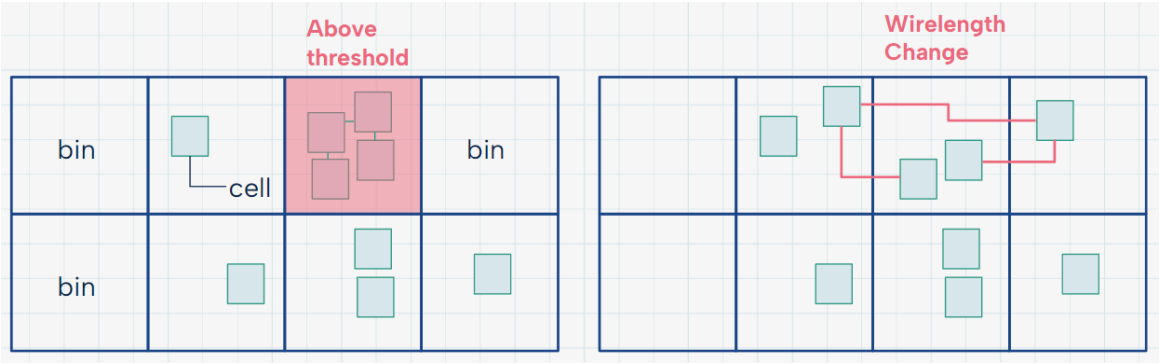


Figure 3: Finetuning based on utilization

2.3.7 Back to clustering with different parameters

After above steps, we iteratively go back to the clustering step with different parameters then perform following steps. The different parameters aim to avoid same clustering results as the previous iteration to expand the search space.

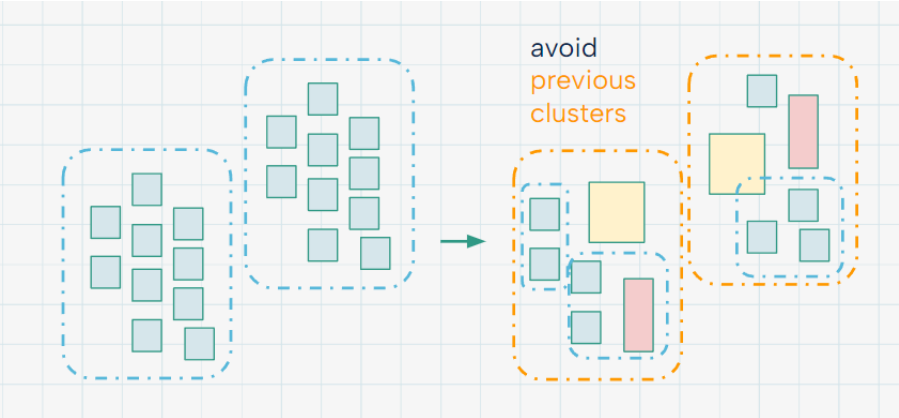


Figure 4: Back to clustering with different parameters

2.4 Parallelization

When designing our method, we carefully considered the potential for parallelization. We will explain the potential for parallelization in each step:

- **Debanking:** This step can be parallelized by debanking different flip-flops in parallel since there is no overlap between different flip-flops in the initial circuit.
- **Force-directed placement:** This step can be parallelized by moving different flip-flops in parallel. **To avoid conflicts, we allow the flip-flops to overlap with each other in this step.**
- **Clustering:** This step can be parallelized by mean shifting different flip-flops in parallel as in the original paper.
- **Greedy banking:** This step can be parallelized by applying the greedy banking algorithm to different clusters in parallel.

Since the runtime of our program in this contest is limited, we want to make the best use of the parallelization to improve the efficiency of our method. But the implementation of parallelization is relatively complex, so we consider to implement it in the clustering and greedy banking steps first, which are the most time-consuming steps in our method.

3 Result

For convinience, we develop a visualization tool to show the placement of cells. In the following figures, the red rectangles represent the flip-flops, the blue rectangles represent the combinational cells, and the shallow blue lines represent the placement rows provided in the testcase. All the wires are not shown in the figures.

3.1 Result of the provided testcase

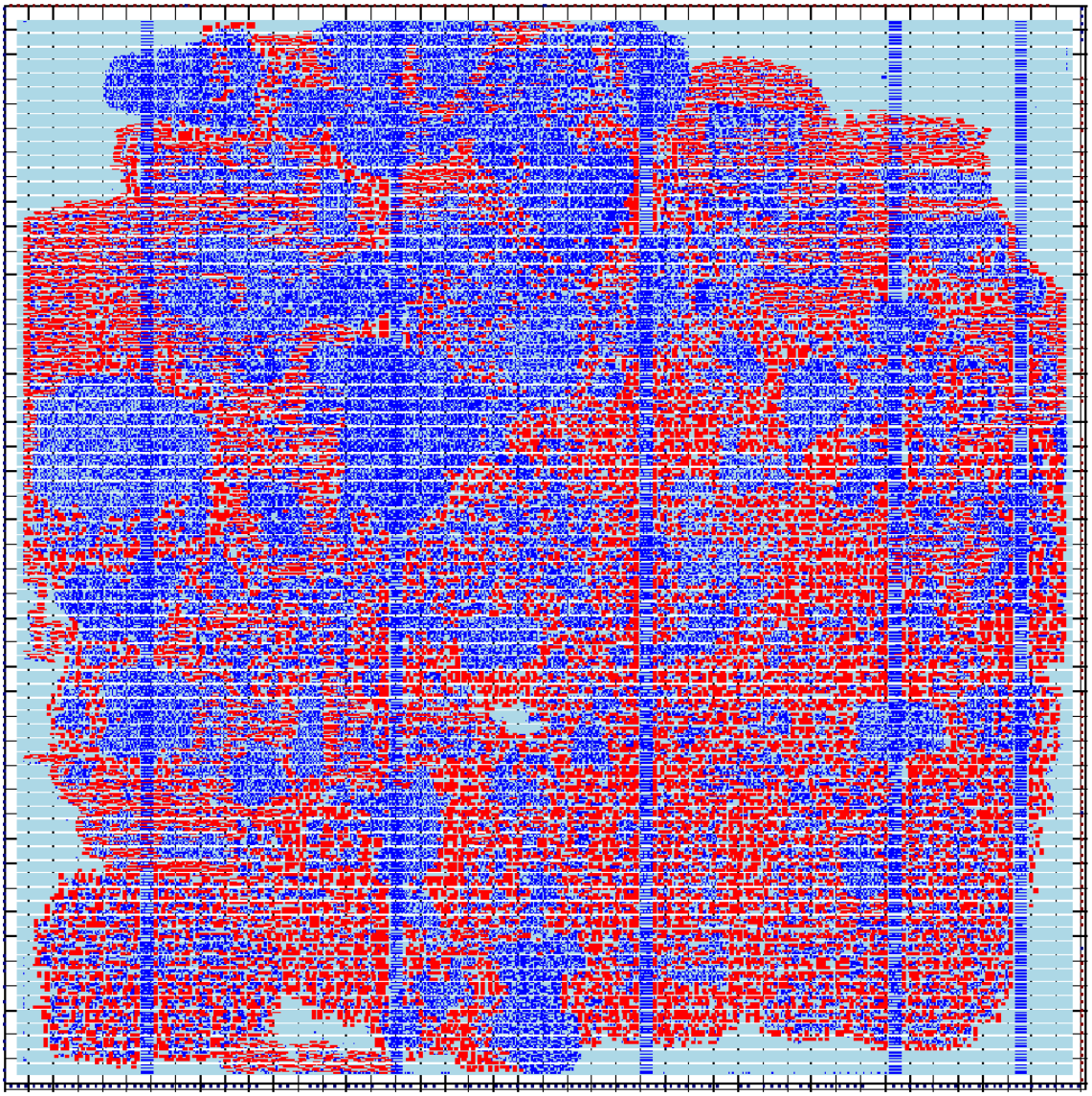


Figure 5: Initial placement

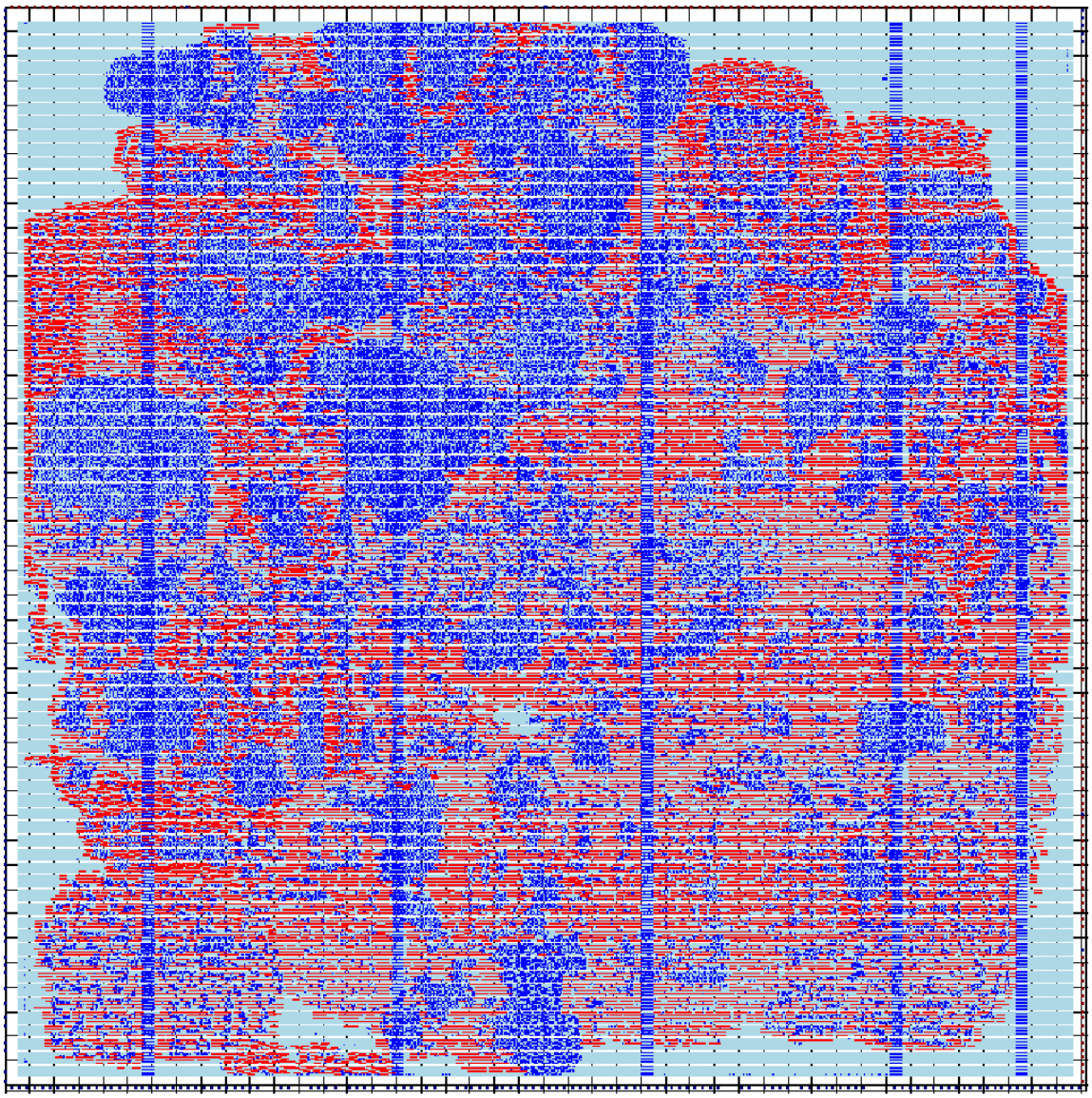


Figure 6: After debanking

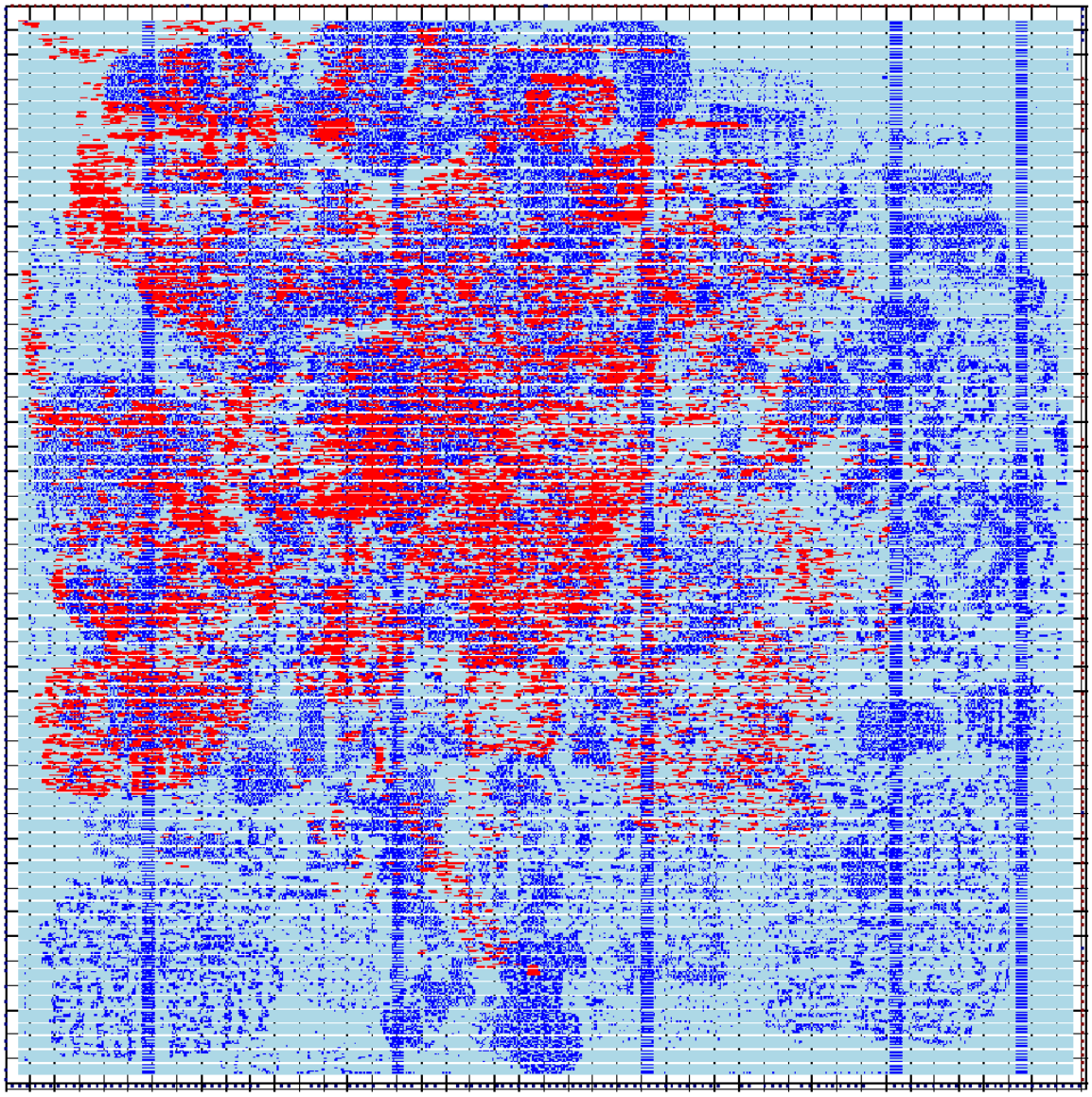


Figure 7: After force-directed placement

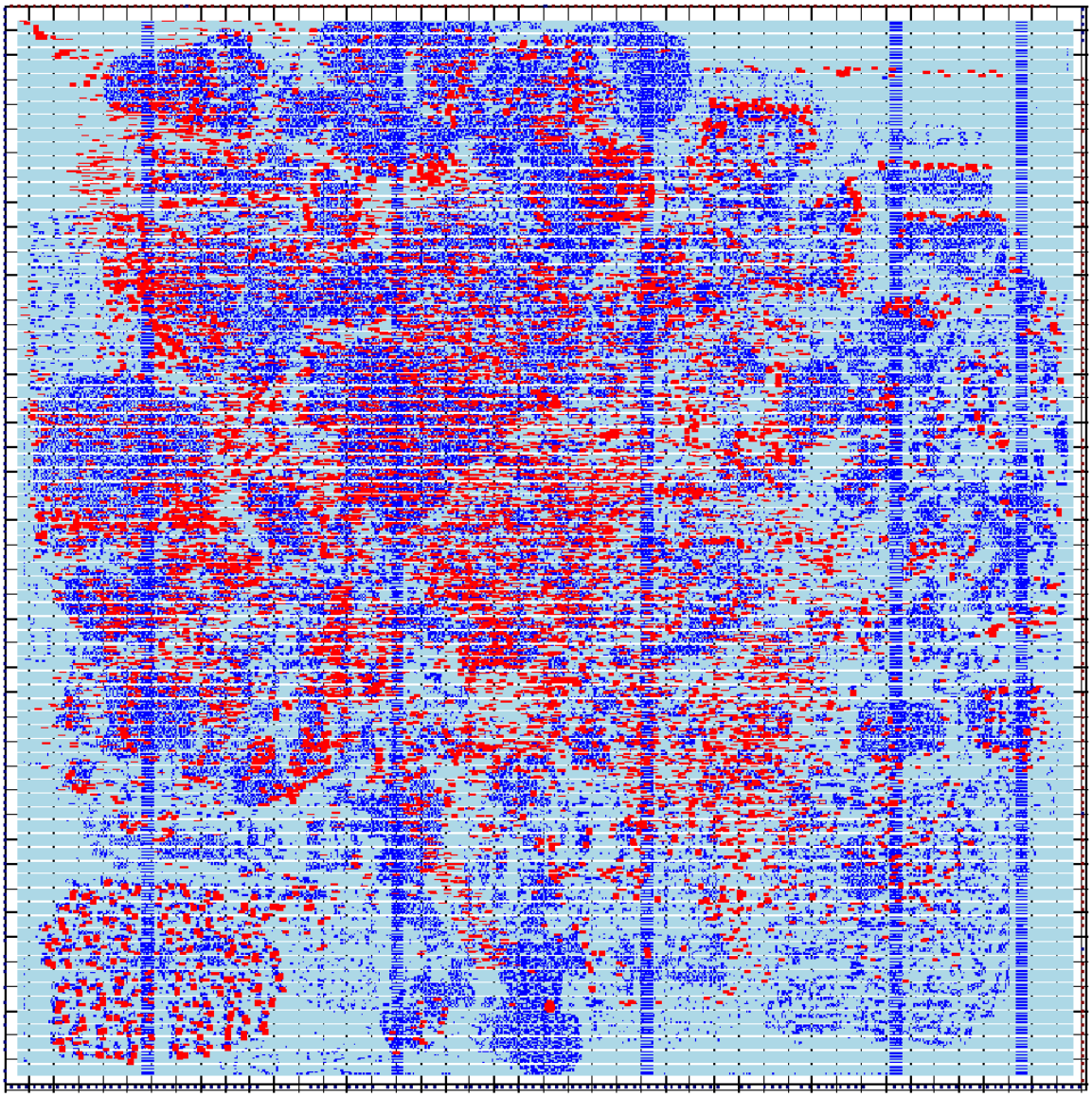


Figure 8: After greedy banking

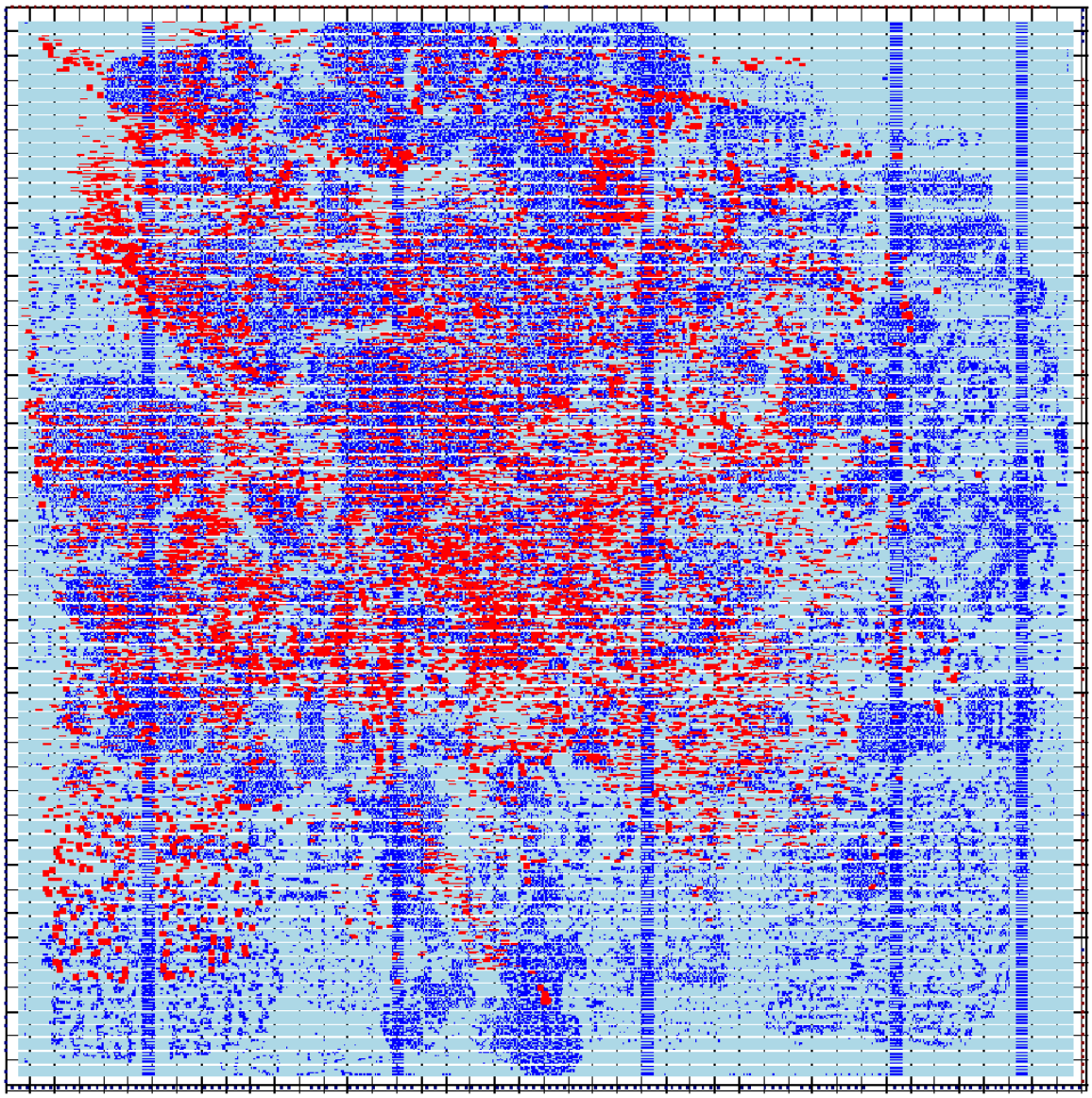


Figure 9: After force-directed placement again

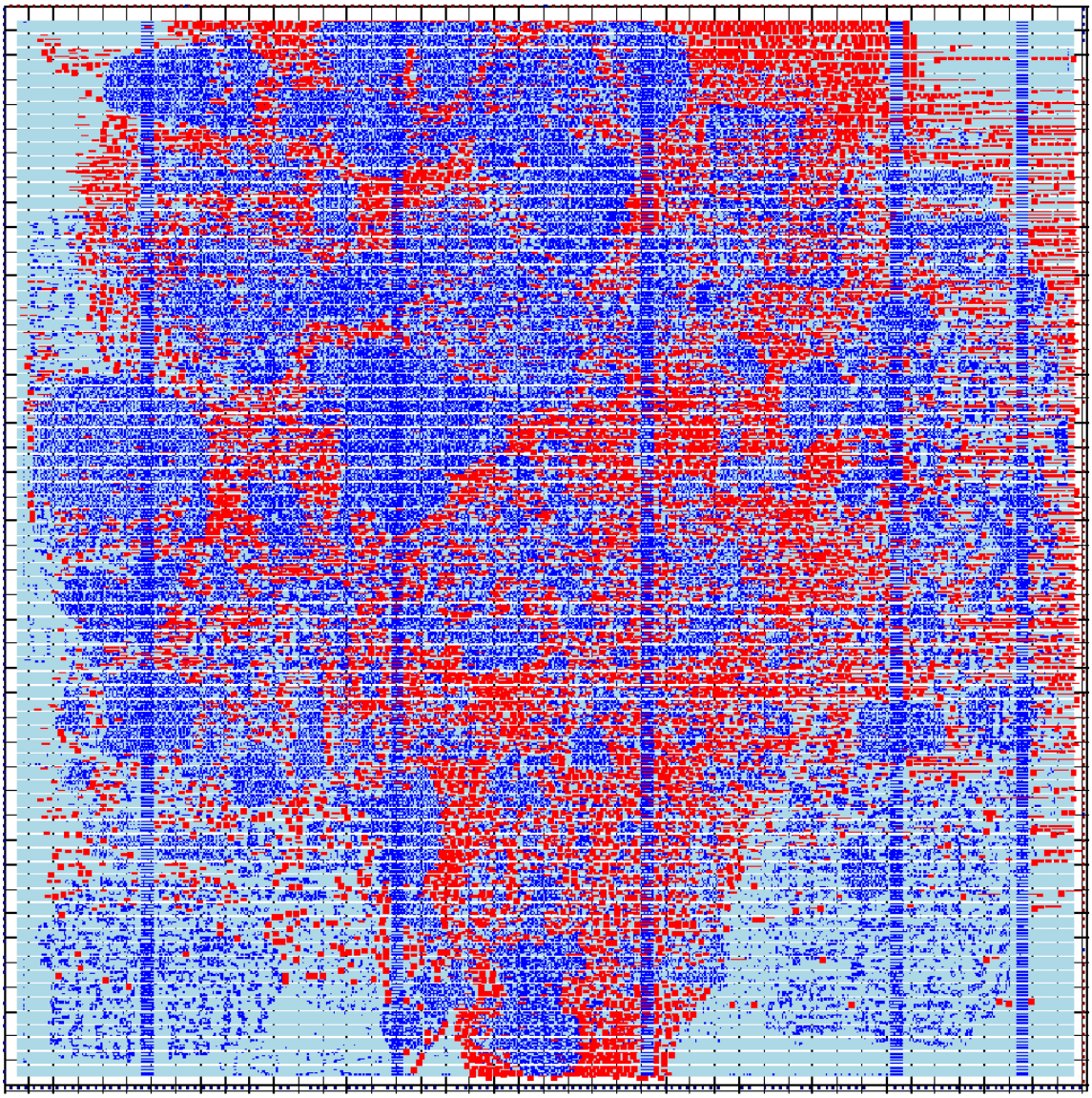


Figure 10: After placement legalization

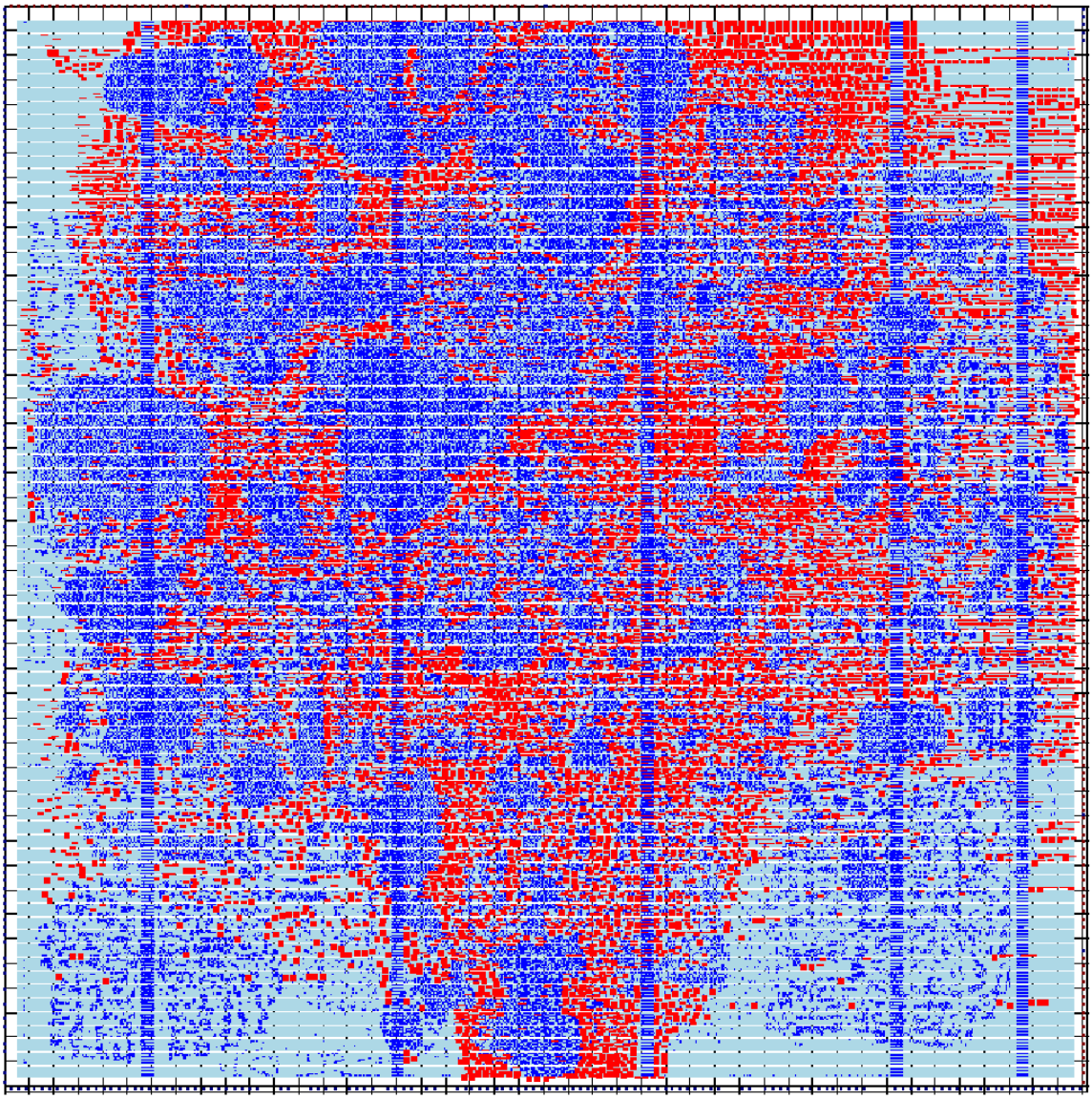


Figure 11: After the second iteration

```
cad main*  ≡
❯ ./sanity cases/testcase1_0614.txt out/testcase1_out.txt
Read testcase ...
Read output ...
Check Inst <name> <libcell> <x> <y> format ...
Check From/pin map To/pin  format ...
Start checking ...
check libType
check pins are in design
check pins are correct connected
check same CLK for banking
check netlist ... step1
check netlist ... step2
Checking pass!
```

Figure 12: Result of the sanity check provided in the contest

3.2 Analysis of the result

As the information of the problem keeps changing, we have a hard time to calculate the overall cost of the result. Though lacking the overall cost, we still analyze the result from the figures above:

- After debanking and the first force-directed placement, the flip-flops are placed in the optimal position for timing as Figure 7. The flip-flops are placed densely, which means the initial placement is a timing compromised placement.
- After greedy banking, the flip-flops are banked together to reduce the power and area cost as Figure 8. There are some multibit flip-flops with larger area generated. Also, there are some large position changes of flip-flops.
- The second force-directed placement optimizes the timing condition of the circuit again. The Figure 9 shows that the position of the flip-flops changes compared to the Figure 8, which means the there are some space to optimize the timing condition in the banking step.
- After placement legalization as Figure 10, the flip-flops are placed in a legal position without overlap. The most significant change from the original placement is the total area of the circuit.
- After the second iteration, there are more flip-flops banked together as Figure 11.

We pass the sanity check provided in the contest, as shown in Figure 12.

4 Conclusion

In this project, we propose a method to solve the problem of "Power and Timing Optimization Using Multibit Flip-Flop". Our method is mainly based on clustering and gain-based greedy algorithm. We think the gain-based greedy algorithm can achieve a good result in this problem as the cost function differs in each testcase. But there are still some works to be done to improve our method, include the calculation of the overall cost and the improvement of the efficiency. Also we need to do more experiments to modify the algorithm to achieve better results after the contest release more testcases and information.

References

[1] Ya-Chu Chang, Tung-Wei Lin, Iris Hui-Ru Jiang, and GiJoon Nam. "Graceful Register Clustering by Effective Mean Shift Algorithm for Power and Timing Balancing." *Proceedings of the 2019 International Symposium on Physical Design (ISPD '19)*, 2019.