PD PA1 Report

Programming Asignment #1 of Physical Design for Nanometer ICs, Spring 2022

系級	學號	姓名	日期
電子所碩一	r10943109	Shiuan-Yun Ding	2022-3-25

Fiduccia-Mattheyses Heuristic

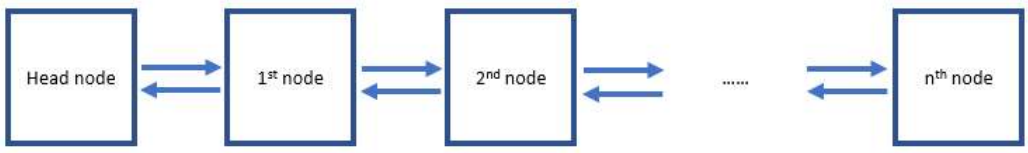
Data Structures

Fiduccia-Mattheyses Heuristic defines a special "bucket list" structures. Since the max gain is stored and updated throughout the whole algorithm, we can get the max gain cell in $O(1)$-time operations.

In my implementation, I defined a bucket list structure `class BucketList` with 3 operations: (1) Insert Node (2) Delete Node (3) Get Max Gain Node.

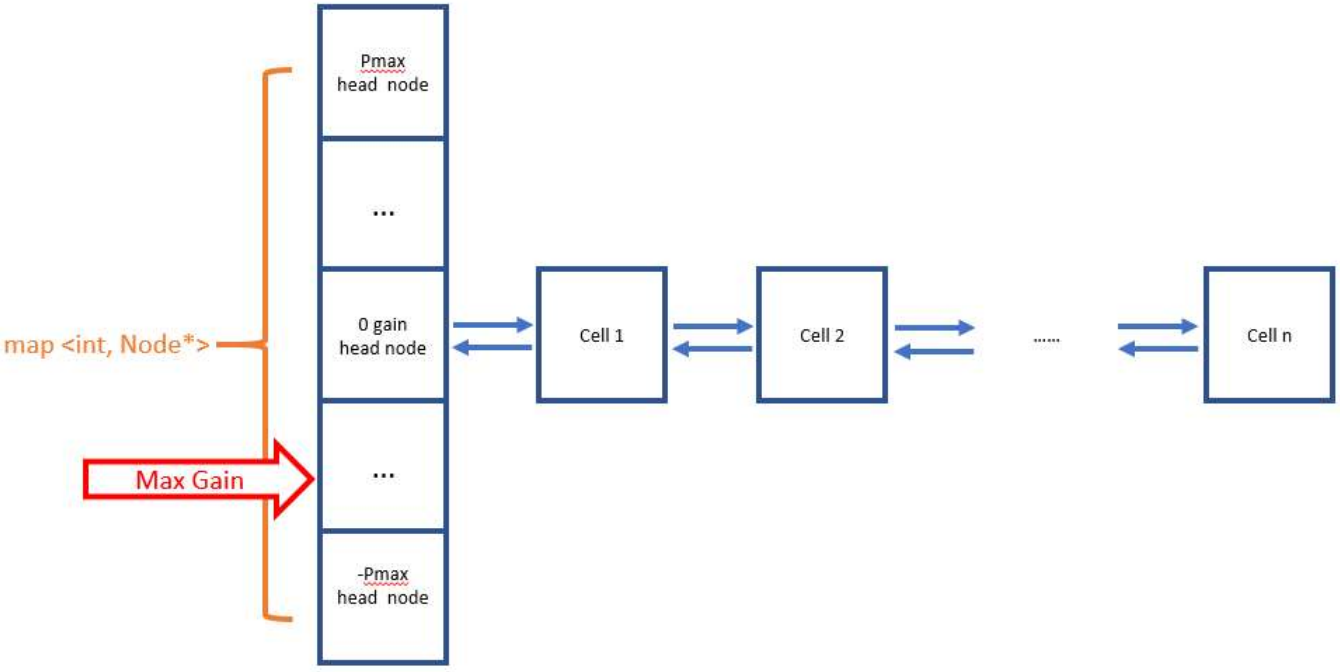
Here are the detailed implementation.

1. Node Structure A simple `class node` structure defined in `src/node.h` has two pointers: one points to its previous node; the other points to its next node. This provides capabilities for double linked list in the further implementation.



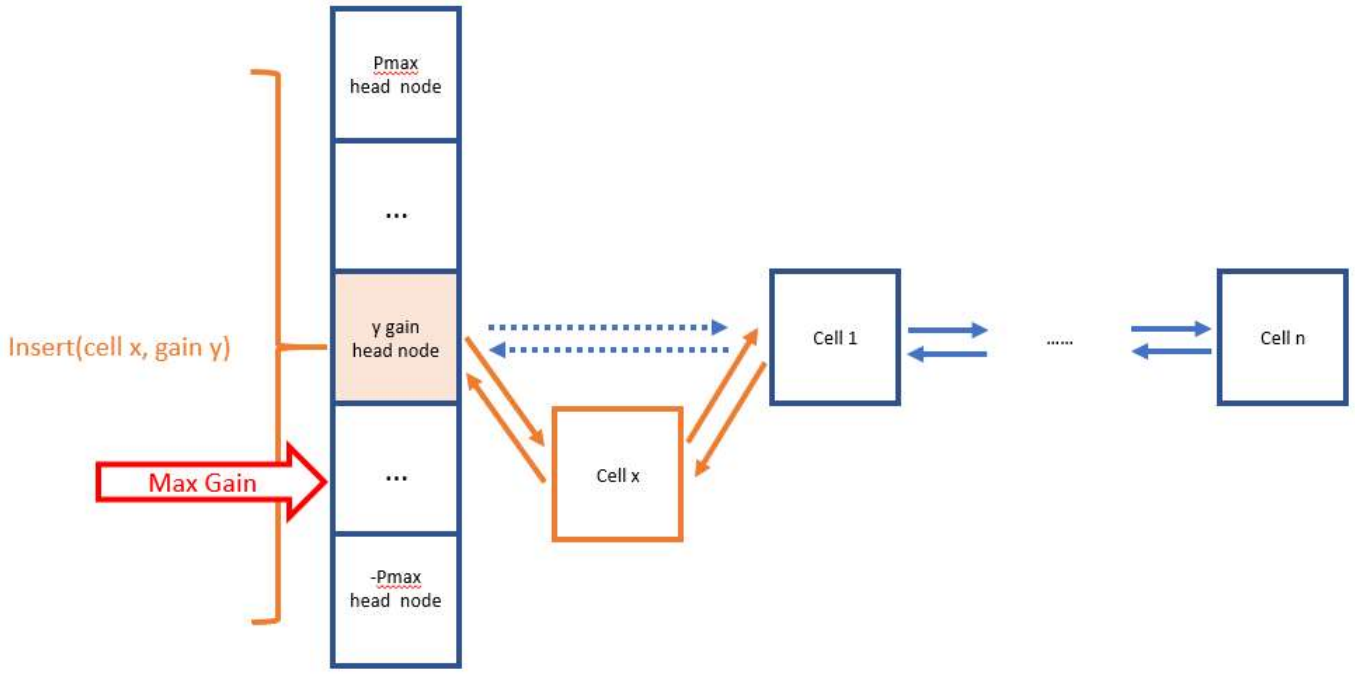
Double linked list structure.

2. Bucket List Structure
   - A special data structure for Fiduccia-Mattheyses heuristic is defined in `src/bl.h` .
   - For each possible gain number, create a head node. Use `std::map` to map gain number and its corresponding head node.
   - Use node structure to connect the same gain cells with double linked list structure.
   - Note that the max gain number is also kept in the bucket list structure.



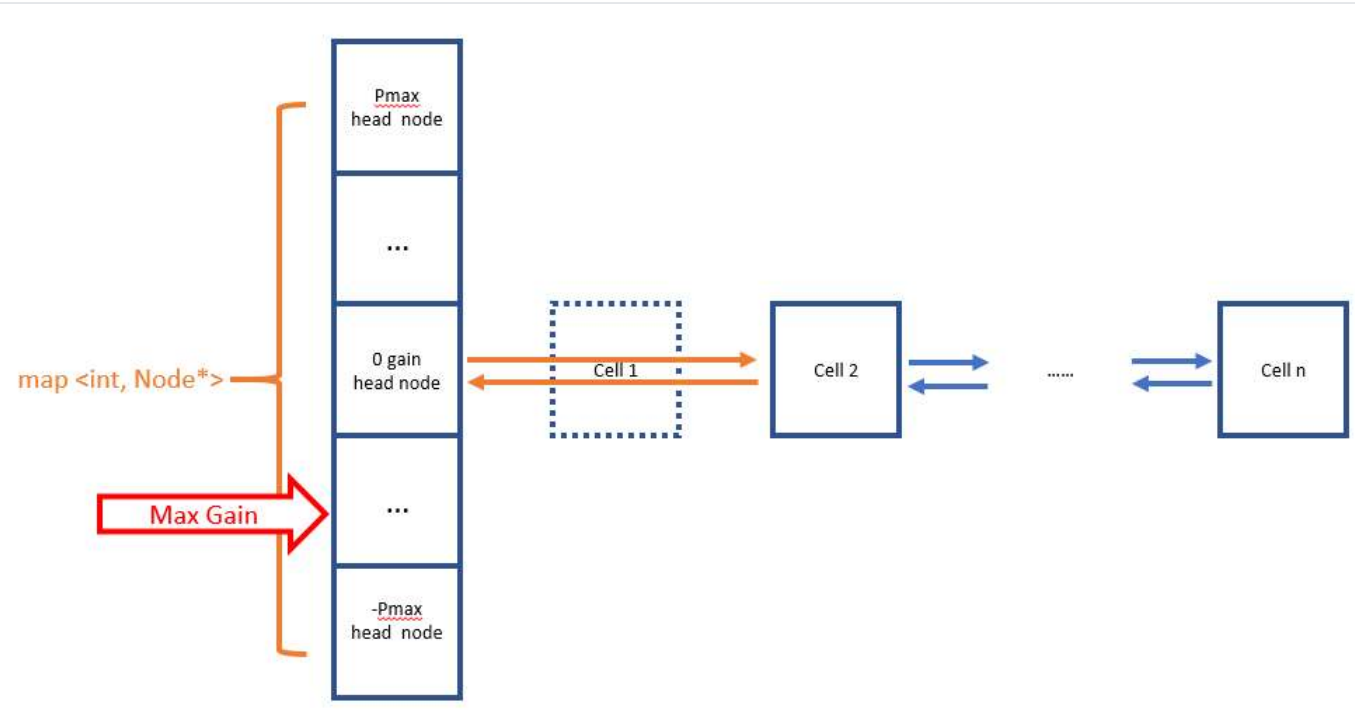
Bucket list structure.
Note that the max gain number is kept.

3. Insert Node Operation
   - The insert operation takes two variables: the cell and its gain.
   - First, use `std::map` to get the corresponding head node. Second, insert the cell between the head node and the original first node.
   - Note that the max gain will be updated if the newly inserted cell has a higher gain.



Insert node operation takes two variables: cell and gain.

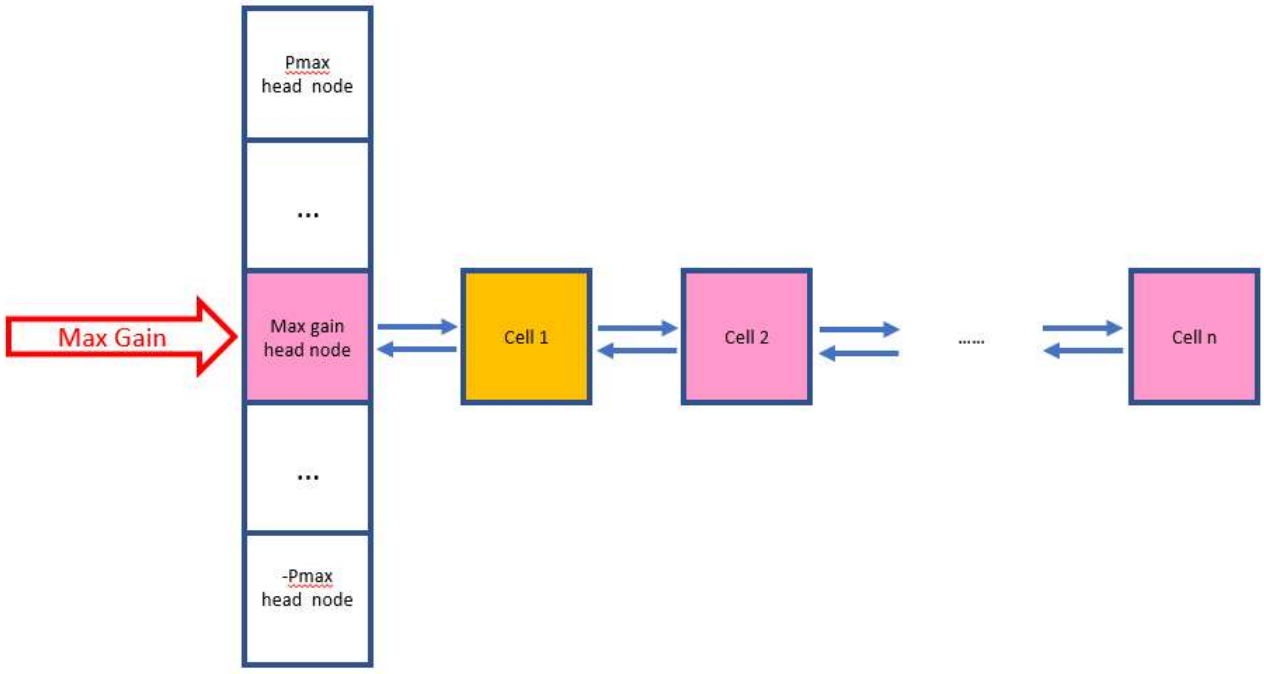
4. Delete Node Operation
   - The delete operation works exactly the same as double linked list.
   - Note that tha max gain will be updated if the deleted node has max gain.



Same procedure as double linked list for delete operation.

5. Get Max Gain Node Operation

- Get the head node corresponding to max gain by `std::map`.
- The first node pointed by the head node will be returned.



The first node (orange node) will be returned.

Findings & Optimization

- Bucket list optimized the timing in a significant way.
- The maximum index corresponding to the maximum partial gain has the largest number in the very first iteration.
  - In the original Fiduccia-Mattheyses Heuristic, for each iteration it will **move all cells** and find the maximum partial gain.
  - Therefore, further timing optimization can be made by setting the the number of **real moves** of the first iteration (suppose it is `index_constrain`) as the constraint and only try to find maximum partial sum within `index_constrain` moves.

Experiment

The `Execution Time` refers to the original Fiduccia-Mattheyses Heuristic; the `Execution Time(opt.)` refers to my futher timing optimization version.

Note that my further optimization does not hurt the performance at all.

Test Case	# of Cells	# of Nets	Initial Cutsize	Final Cutsize	Execution Time	Execution Time(opt.)
input_0	150750	166998	65799	12742	2.02 sec	0.78 sec
input_1	3000	5000	2400	1692	0.00 sec	0.00 sec
input_2	7000	10000	4470	2198	0.06 sec	0.04 sec
input_3	66666	88888	47238	27845	7.19 sec	7.12 sec
input_4	150750	166998	82801	45045	38.02 sec	26.03 sec
input_5	382489	483599	251653	143932	210.19 sec	171.71 sec