A Fast Macro Placement Methodology Based on the Logic Group

Albred Mo
Physical Design Engineer
Nvidia

March 23, 2010
Renaissance Shanghai Zhongshan Park Hotel
Shanghai, China

A Fast Marco Placement Methodology Based on Logic Group

Abstract:

The paper introduces a fast floorplan method for high macro count and hard timing design. Magma talus can generate a quick placement for all macros and standard cells based on logic group, then we can use this placement info to highlight and divide appropriate "macro group", which is faster and more reasonable than normal group method for high macro count design. For hard timing design, we can use aggressive "macro placement" method to show the critical logic group then tune the placement shape (keep critical logic close together without macro and other logic group blocking) by incremental "macro placement". This method can lead to better timing (both WNS/TNS) and better routing result (total wire length) for our regular design.

Key Words: floorplan, macro placement, logic group, coarse placement

1. Introduce

Generally, placing macros is the major work for the floorplan for a block level design. Traditionally, our initial macro placement may base on the previous floorplan style; we should consider IO connection, and we need to know the logic relationship between macros and IO, then place macros near to related IO; we trend to place macros around the boundary; we'd like to place the macros with the similar name pattern together; we run placement (fix_cell), then we tune the macros placement based on critical timing path or congestion, and we may show the flyline from/to macros to see if the macros and std cells are placed reasonably…

For the traditional floorplan method, we may take long time to know our design, may discuss with front end designer, and may take many physical placement iterations to close timing. It will be a terrible work for a design with large macro count.

Now we have a new floorplan method based on logic group which can close floorplan job more quickly and more effectively. The flow has 3 steps:

1) place macros and std cells at the same time (coarse placement, "run place cluster");
2) divide logic group (both for macros and std cells);
3) Incremental placement, tune the group shape and location.

2. Coarse placement

We can divide a full placement work (fix_cell) into two steps:

1) Coarse placement: performs initial placement for the design;
2) Incremental optimization based on the coarse placement.

Step 1) is the key point to get a good placement result, and our discussion will be focused on coarse placement.

There are two different "coarse placement" commands in talus –"run place cluster" and "run place global". Both these two commands can do coarse placement based on logic group (cluster). And "run place cluster" can place macros and std cells at the same time, while "run place global" can do std cell placement with congestion consideration. As we known, "fix_cell" uses "place global" to do coarse placement. According to our test, given the same macros location, the placement result is very similar for "place global" and "place cluster". So we can predict coarse placement result of fix_cell based "place cluster" result.

The picture shows the "place cluster" result for design case D1 (**Fig. 1**):

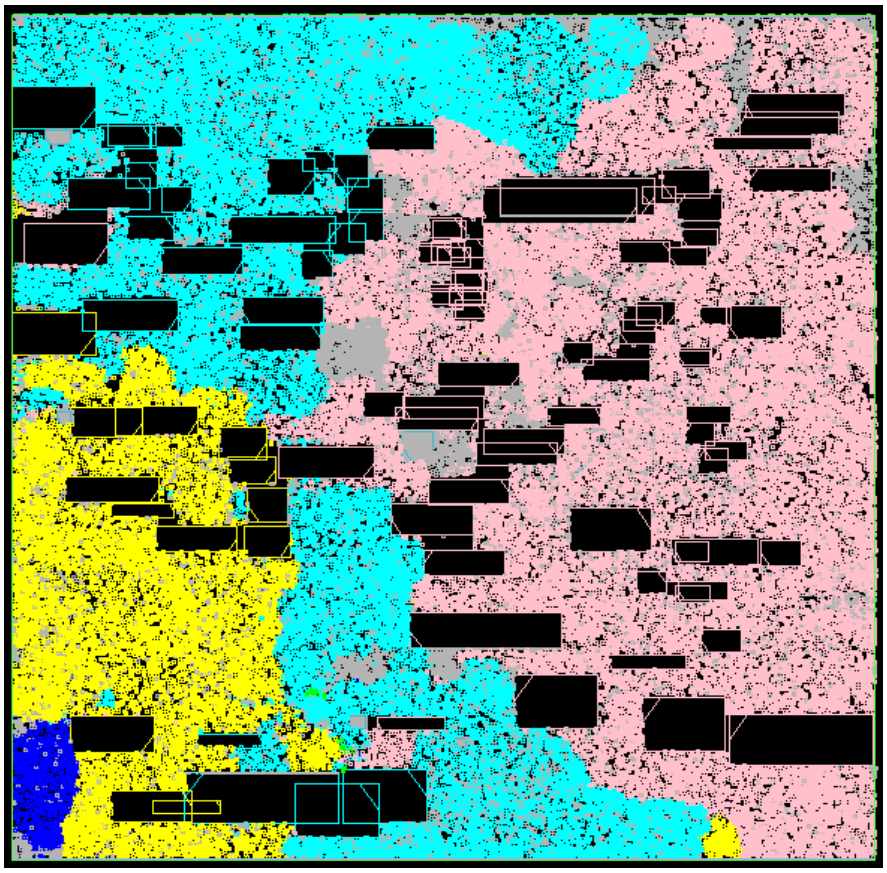

Fig. 1 “place cluster” result (coarse placement)

The full command we recommend: “run place cluster $m –effort high –macro_style overlapping”. With macro overlapping allowable, we can divide macro group more easily. If you don’t add option “-ignore_io”, it will consider IO connection by default. And the runtime is 5~10 minutes for D1(600k instance, “place global” will consume about 2~4X times).

Instead of timing and congestion, the most important cost point for coarse placement is wire length (**Fig. 2**).

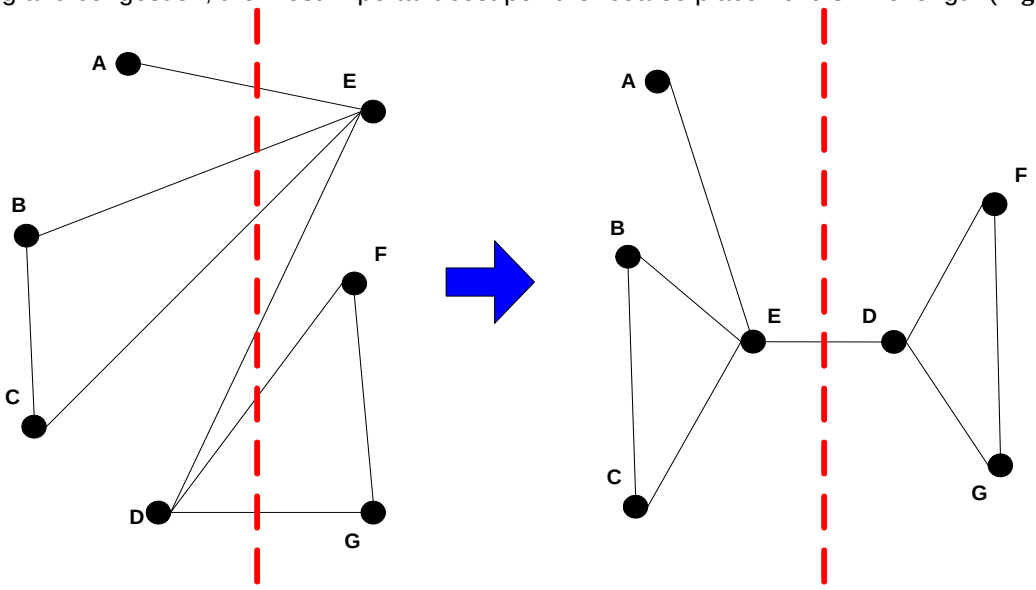

Fig. 2 Minimize the wire length

For the left placement style, it groups (A, B, C, D) and (E, F, G), then place. For the right placement style (wire length based placement), the placer places the cells with much connection to each other together ((A, B, C, E) and (D, F, G)), which leads to minimum total wire length. Then we can see that the flyline across these two groups is less than the left one, which is also good for congestion. So we should divide logic group as placer do to get a predictable placement result.

3. Divide logic group

There are some principles for dividing logic group.

1) Grouping macro and the logic connected to it together. As we know, a macro may have lots of input/output pins and have busy communication with the logic connected to it, and the location of the macro will highly affect the placement of the logic, so we should group the macro and the logic together, and there should not be flyline directly into (out from) macro across different group.
2) There should be as fewer flylines across different groups as possible. In other words, if you see many flylines across two groups, then they should be group together.

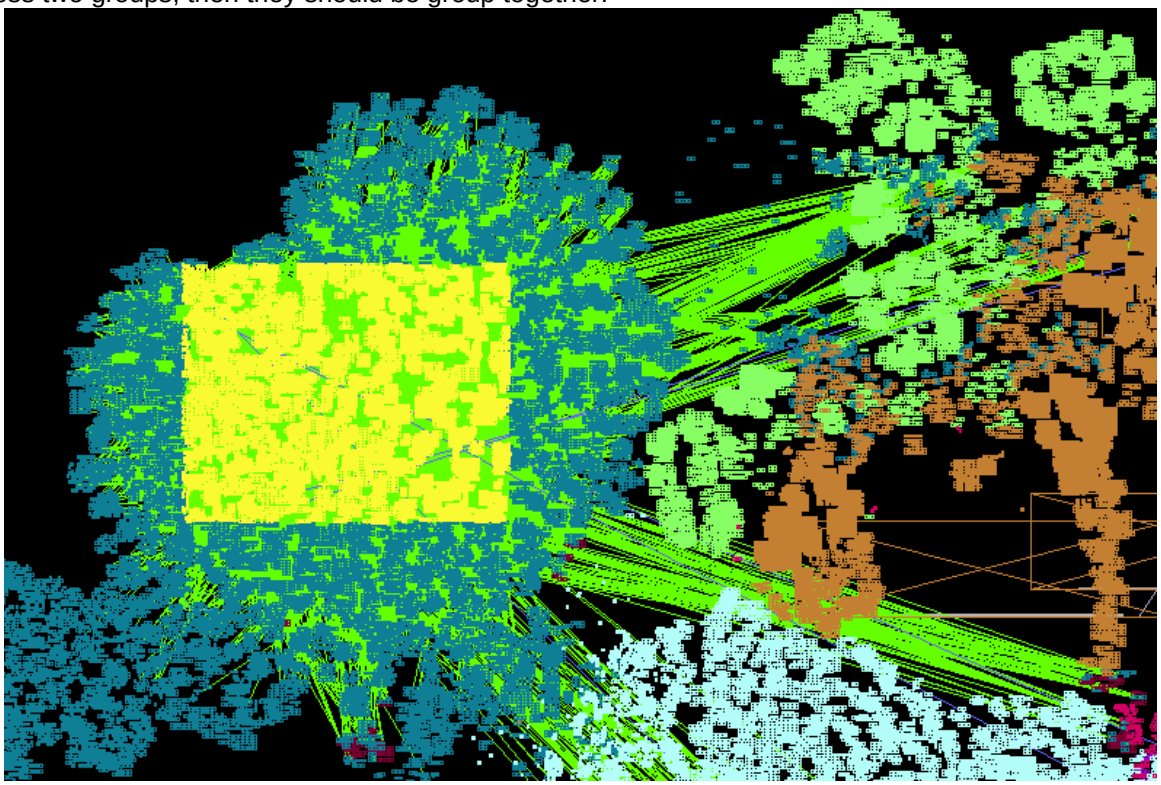

Fig. 3 Flyline of some selected cells in a group

As the **Fig. 3** shows, most flyline are inside the “dark green” group.

For the easy design that has good hierarchically divided in RTL, we can use hierarchy browser in talus to divide logic group.

Take D1 for example, the hierarchy naming style is “AAA/BBB/CCC_1/…”. We can easily divide D1 to 3 logic groups, then place macros. Here is the implement bellow (**Fig. 4**):

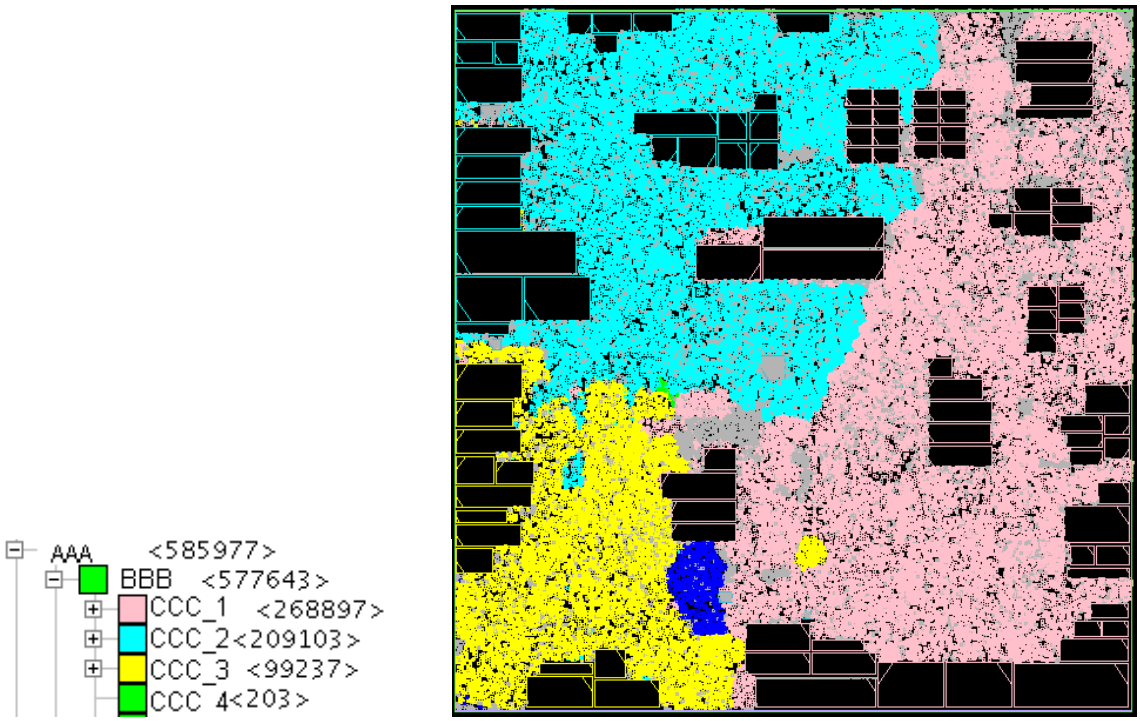

Fig. 4 Groups with different color using hierarchy viewer

We can collapse the hierarchy tree more deeply to get more detailed group division (**Fig. 5**):

Fig. 5 Groups with more detailed hierarchy collapse

For some design whose name style is not so friendly, we can't get clear group info from the hierarchy browser. Then we should improve our "coarse placement" with extra option: "run place cluster $m –effort high –macro_style overlapping **–max_util 1"**.

The default max_util is 0.6, and the lower the value, the more the group clusters are spread out over the available area. So we set “max_util” to “1” to make the group cluster more concentrated, and it’ll be much easier to divide logic group.

Take another design D2 for example.

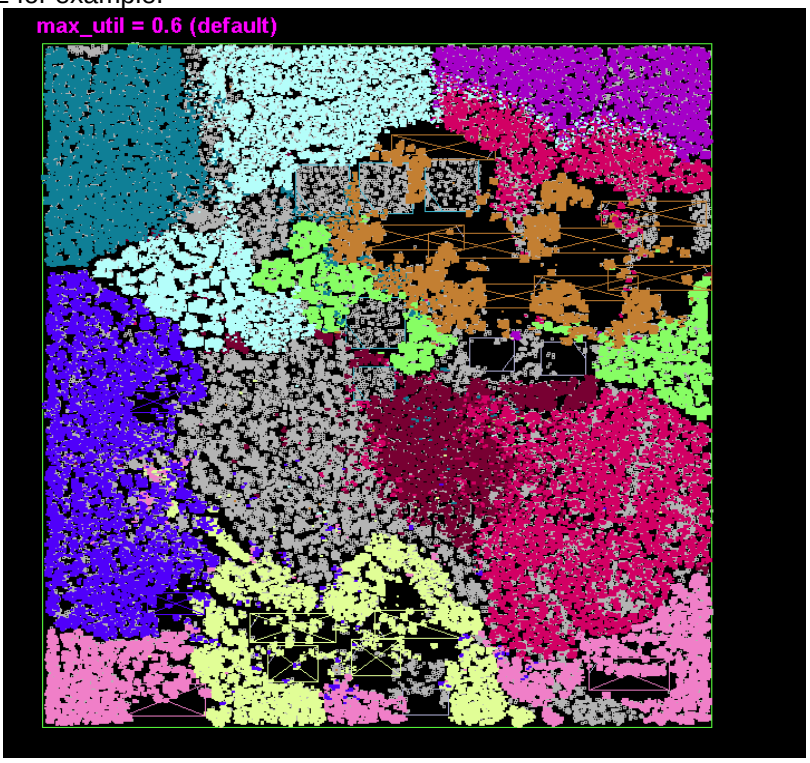

Fig. 6 Default “place cluster” result (max_util=0.6)

The **Fig. 6** shows the result of “max_util” = 0.6 (default). We can see that it spread out the std cells all over the chip.

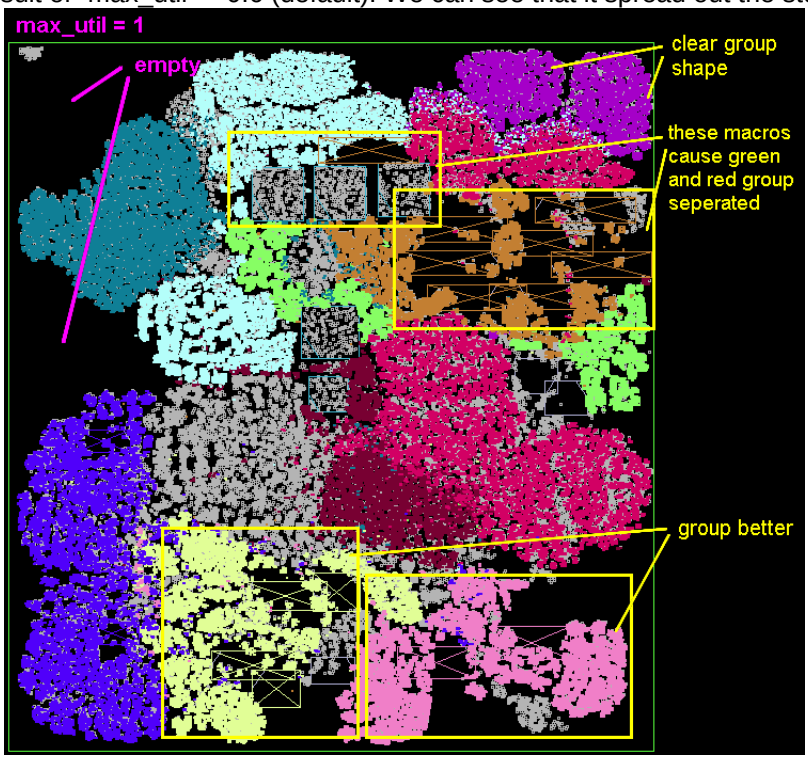

Fig. 7 “place cluster” with “max_util=1”

When we set max_util = 1(**Fig. 7**), the group cluster are placed more concentrated. There is much empty space in the chip, and it'll be more flexible for the placer to place the group cluster. We can see that the group "yellow" and the group "pink" has better group shape than before. But we still see that group "green" and "red" are separated by some macros. We should use more aggressive method to solve this problem: we can stack those macros together and freeze them, then "run place cluster" again (it won't touch fixed cells) (**Fig. 8**).

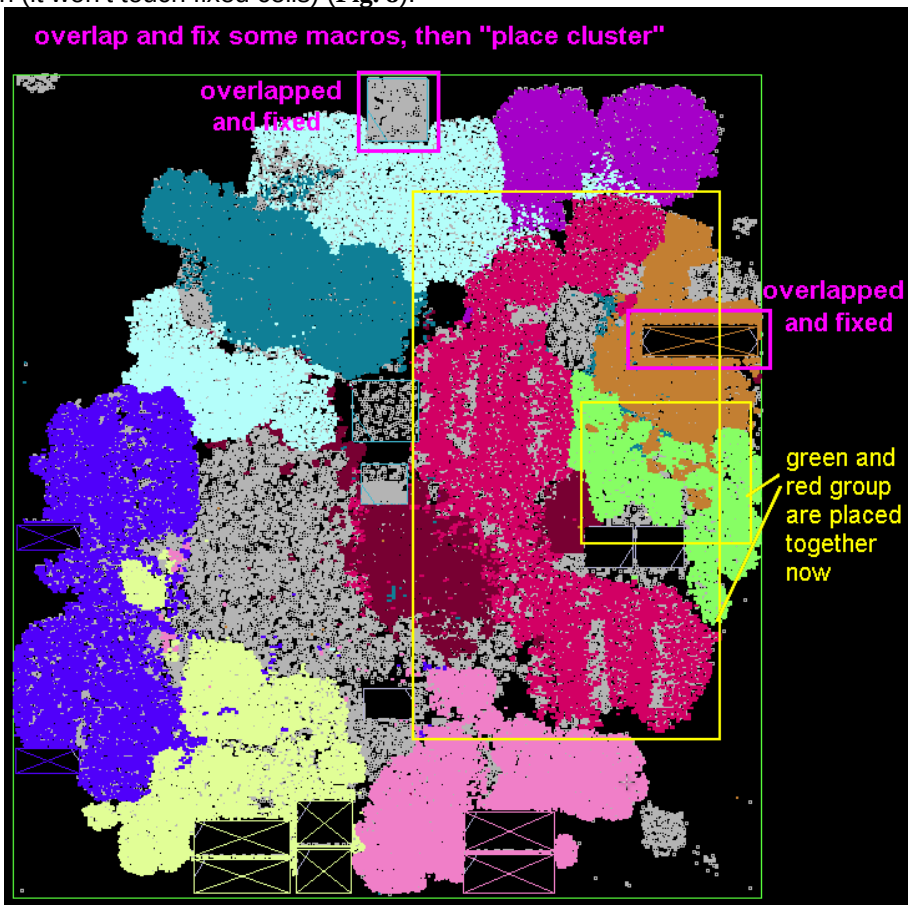

Fig. 8 Stack and fix some macros to free more placement space

Now the group "green" and "red" look much better. It proves that they're real logic group and should be placed together.

4. Tune the group shape and location

When we have clear logic group division, we can place the macro near to its original location of "coarse placement". Then we fix those macros and run "place cluster" again. The placer will place std cells only this time. According to the placement result, we will try to modify the macro location to tune the logic group shape and location, especially for timing critical logic group. The criterion of good group shape is:

1) Placed like a "pie";
2) No std cell logic of other logic group invading the "pie";
3) No big macros inside the "pie".

The "timing critical group" should be defined as the group which has lots of failing end points (flip flops) and contributes most part of total negative slack, instead of only several flailing end points in the group which may have the worst negative slack value. Generally, the start points of these paths are also in this group; these paths have deep logic level and the start point may have high fanout end points (FFs). According to our group dividing principle, to minimize the flyline cut count across groups, these start and end points should also be grouped together. Then we should place this group like a "pie" which makes the logic cell inside more tightly and minimizes the path delay (fewer buffers inserted and soless area).

Take design D3 for example (**Fig. 9**).

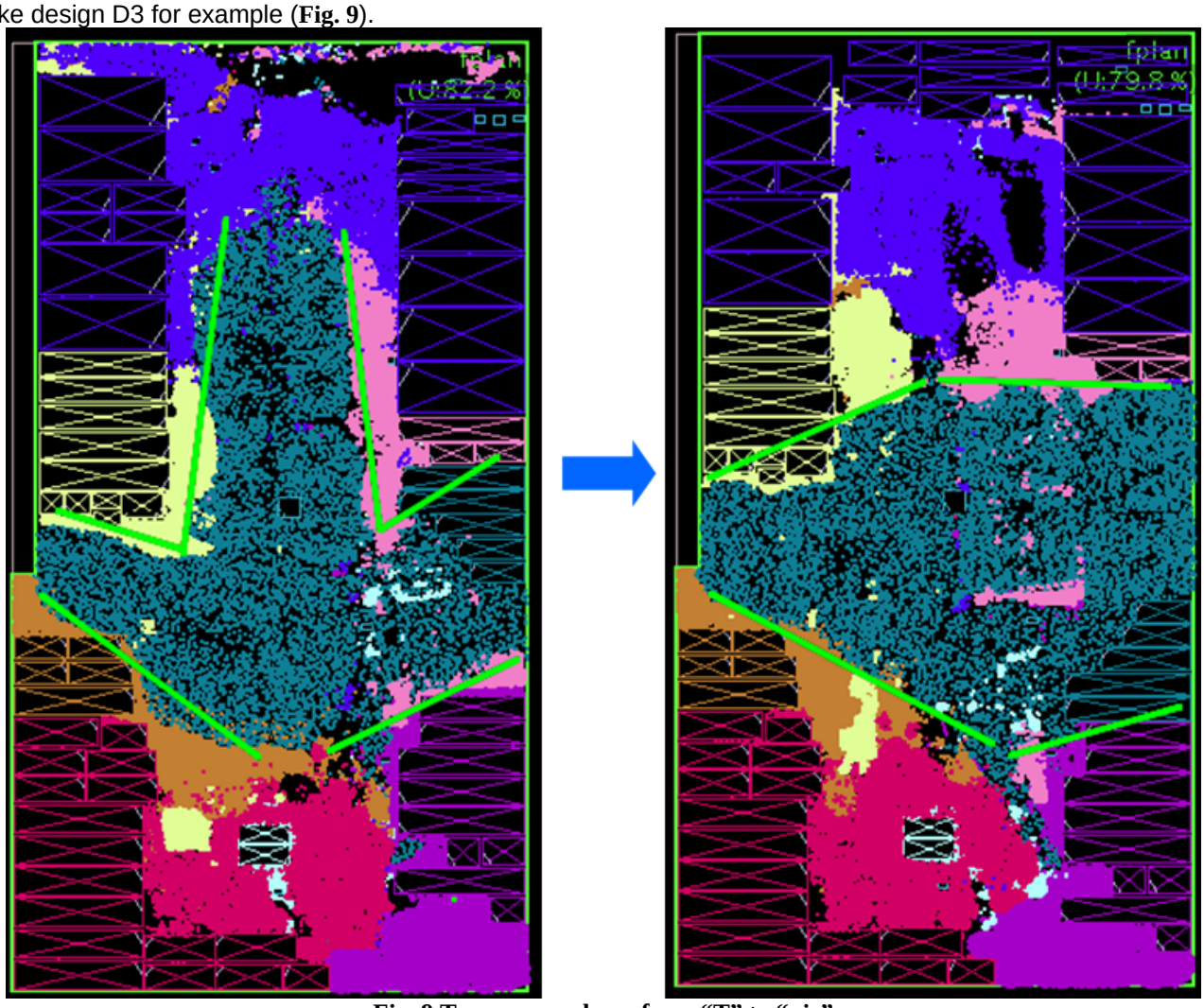

Fig. 9 Tune group shape from “T” to “pie”

The group “dark green” is the timing critical group, and is placed as “T” down shape in the left floorplan. We tune the macro placement, and make the group placed as a “pie” shape in the right floorplan. The timing gets much better.

Sometimes, we need to clean some space to get good shape of the critical group, so we may move other noncritical group out of the related region. One thing we can do is tuning macro location. When you move the macro away, the logic connected to the macro will follow it. In fact they are the same group, so moving macro can tuning group location and shape.

Here is an advanced application example (**Fig. 10**).

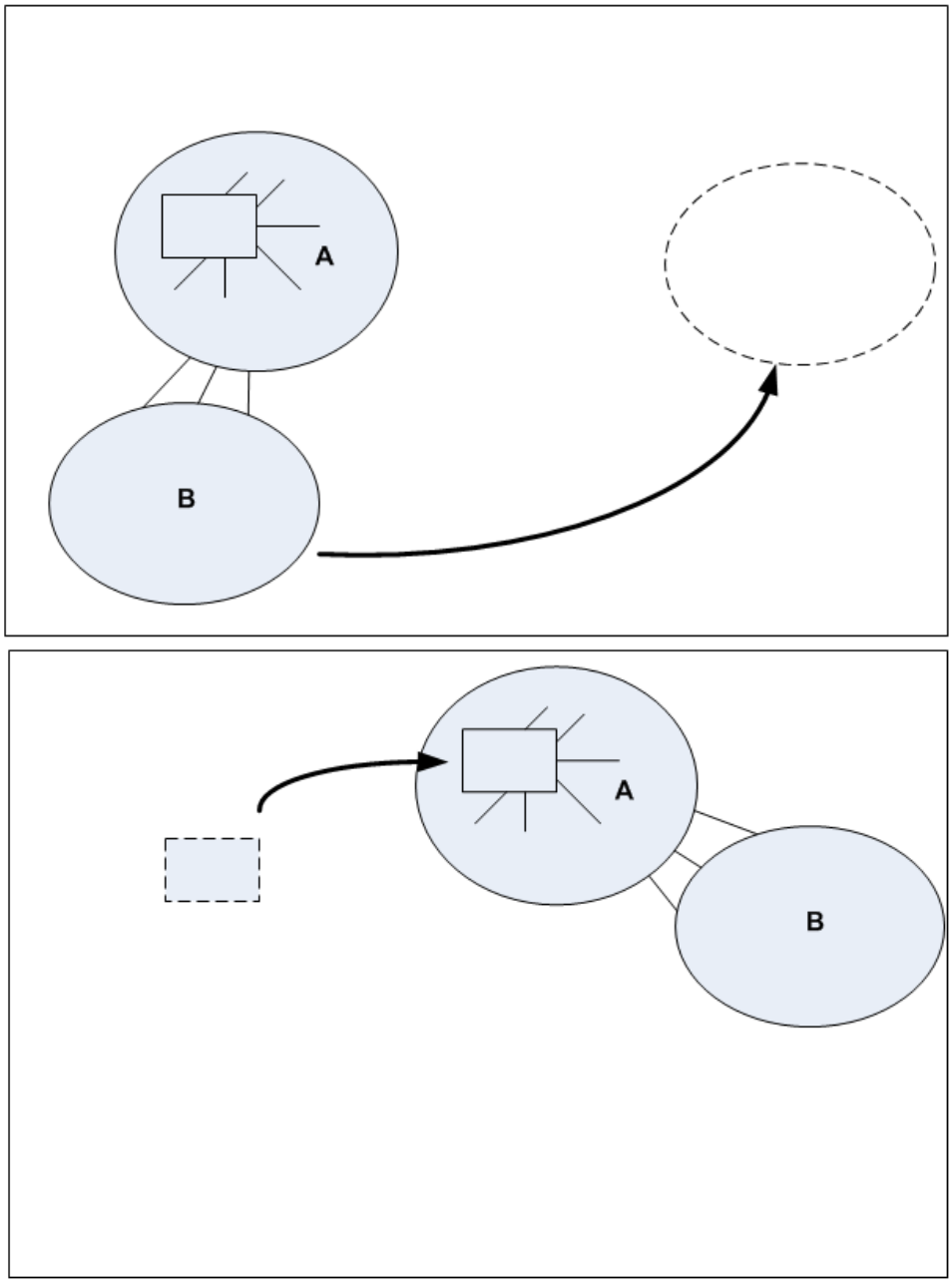

Fig. 10 Drag group by moving macro

There is a macro in group A, which is strongly coupled with group B, and we want to move group B away from the left bottom corner to the right of the chip. Though there is no macro in group B, we can move the macro in group A and then the group A will drag group B to the right side.

Sometimes we can't put all macros around the boundary, for example, there are lots of macros occupying large area of the chip, or the pin density is very high at boundary and placing macros near the boundary may cause serious congestion problem. In fact, we can put macros all over the chip, of course, with a reasonable way:

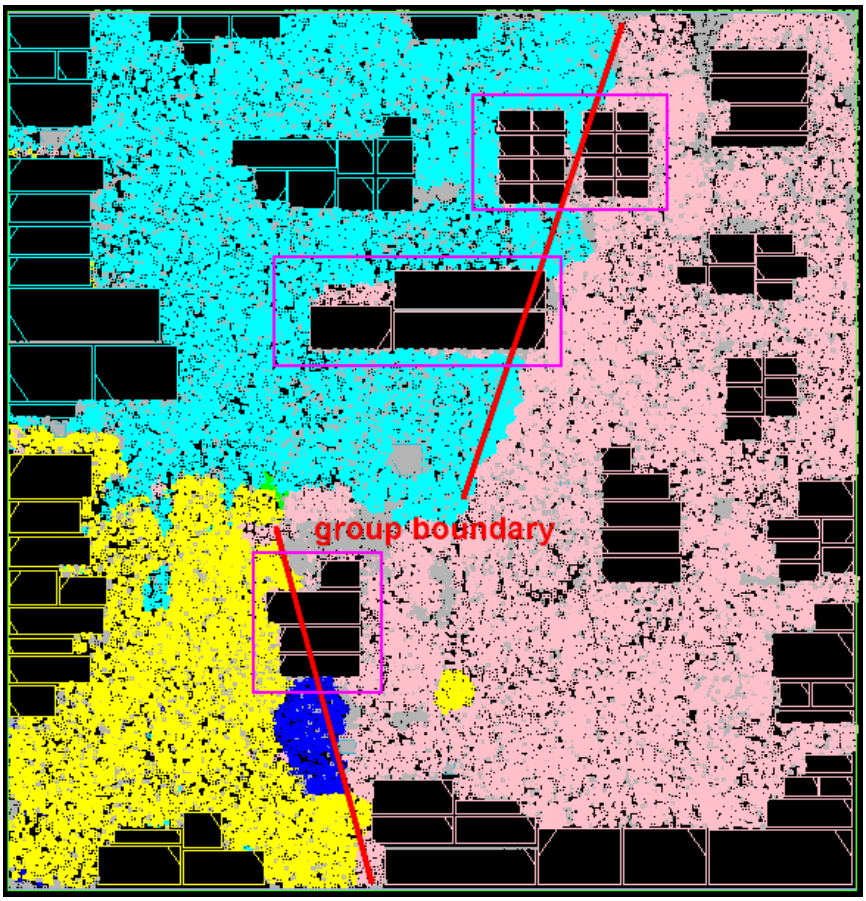

Fig. 11 Place macros at group boundary

As **Fig. 11** show, we put some macros of the group “pink” at the group boundary, which makes the flyline cut across these macros as fewer as possible and doesn’t affect the group shape too much for both group “pink” and other groups. You can suppose that each group is a lower level partition, and we just place macros along the partition boundary.

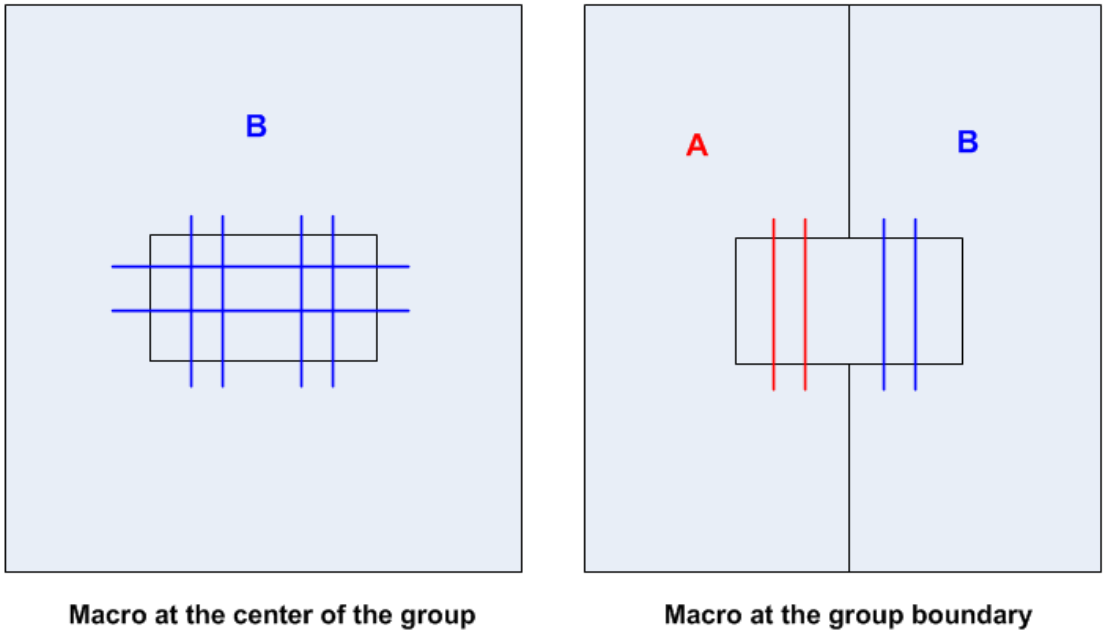

Fig. 12 Macro at the center VS macro at group boundary

As we know, if we place a macro at the center of the logic group, there are many flylines of this group across the macro, both vertical and horizontal as the left picture. Even worse, the macro will occupy the “good” position which makes std cells of this logic group can’t be placed tightly and adds extra detour path delay. While we place the macro at the group boundary, the flyline across the macro will be only vertical and no horizontal as the right picture, because there should be few flylines across different groups. And it will make less impact to std cell placement for group B. The expense is that the macro of the group B intrudes the space of the group A. So you should make a choice that which group has the higher priority.

All these tuning trial can be easily and quick applied by “place cluster”. You can move and fix some macros (“place cluster” doesn’t touch the fixed cell), then run coarse placement to see how the group’s shape and location change, which looks like “continental drift“. It’ll save lots of time without running full fix_cell trial.

5. conclusion

Placing macro based on logic group with quick implement by “run place cluster” is a fast and effective method to handle our regular floorplan work. Total wire length, routing congestion, area and timing (WNS/TNS) are not conflict goals with a good macro placement.

We can use “place cluster” to go though most of the trials at the beginning and use “place global” to get more accurate placement result at the end.

In our opinion, a design is like a house: logic groups are different rooms (bedroom, kitchen, living room…), and we hope these rooms have good division and regular shape; each room has its separated function, when we have lots of guests in the living rooms, they won’t go to our bedroom or kitchen too often (minimize cross cut); macro is the furniture in the room, generally we should put the furniture standing against the wall and they won’t block our movement; we shouldn’t put them against the window (high pin density), which would block the sunlight: