Stage 3 – Floorplan

Team info :

- Sara Abo Salah 208298562 , username : saraabosalah , Saraabosalah@mail.tau.ac.il
- Anan Break 322663170 , username ananbreak , ananbreak@mail.tau.ac.il
- Adam Salameh 20753538, username Adamsalameh , Adamsalameh@mail.tau.ac.il
- Nawaf Gadban 314702705 , username: Nawafgadban , Nawafgadban@mail.tau.ac.il
- Lama Metanis 314857152 , username: Lamametanis , Lamametanis@mail.tau.ac.il

Workarea : /project/advvlsi/users/saraabosalah/ws/stage3/workspace/stage3

3.1 Warmup Questions :

- [#P3.1_Q1] Why is the floorplan stage important in physical design?
    - Area Optimization: During the floorplan stage, the layout and placement of various functional blocks and components of a chip or integrated circuit (IC) are determined. Efficient floorplanning techniques help minimize wasted space and maximize the utilization of the available area, leading to cost-effective designs.

    - Performance Optimization: The floorplan stage plays a significant role in achieving the desired performance of the chip. The placement of critical components, such as processors, memories, and input/output (I/O) interfaces, can impact the overall performance.

    - Power Optimization: Proper floorplanning can help optimize power consumption in a chip. By considering factors like power supply networks and thermal issues, designers can strategically place power-hungry blocks and ensure efficient power delivery. This includes minimizing power dissipation, reducing voltage drops, and optimizing power routing to minimize overall power consumption and improve energy efficiency.

    - Design Closure and Iteration: The floorplan stage sets the foundation for subsequent design stages, such as placement and routing. By achieving a robust floorplan, designers can estimate and allocate resources effectively, leading to smoother design closure and reduced iterations. It allows for efficient utilization of available routing resources and reduces the chances of congestions or timing violations during later stages.

  The floorplan stage serves as a critical early step in physical design, enabling to optimize area, performance, and power of the chip.

- [#P3.1_Q2] What are some of the factors to consider when creating a floorplan?
    - Block Placement: The placement of functional blocks, such as processors, memories, I/O placement is critical. Factors like block size, connectivity requirements, power consumption, and heat dissipation should be considered when determining their optimal placement. Placing related blocks closer together can minimize signal delays and improve performance.

    - Power Distribution: Efficient power distribution is crucial for proper functioning of the chip. The floorplan should consider the placement of power supplies, power routing, and decoupling capacitors. Balancing

power supply network to minimize voltage drops and ensuring sufficient power delivery to different blocks are important considerations.

- I/O Placement: The positioning of I/O (input/output) placement is significant for efficient data transfer between the chip and external devices. The floorplan should consider factors such as signal integrity, power distribution, and the physical constraints of the package or board where the chip will be integrated.

- Timing and Performance: The floorplan should take into account the timing requirements and performance goals of the chip design.

- Area Utilization: Optimizing the utilization of available chip area is essential to minimize the cost and improve its efficiency. The floorplan should aim to maximize the usage of available space, avoiding unnecessary gaps or wasted area. Efficient placement of blocks, power supplies, and routing resources can contribute to better area utilization.

By considering these factors and striking a balance between all the requirements, we have to create an optimized floorplan that lays the foundation for a successful chip design.

- [#P3.1_Q3] Congestion:
  What is congestion in a floorplan ?
  Congestion in a floorplan refers to the situation where there is an excessive concentration or congestion of routing resources, such as wires or metal tracks, in a specific area of the chip layout. It occurs when the demand for routing resources exceeds the available capacity, leading to difficulties in completing the necessary connections between different blocks or components.

  Why is congestion a concern in floorplanning?
  - Routing Difficulties: Congestion restricts the availability of routing resources, making it challenging to establish proper connections between different blocks and components. The limited availability of routing channels and metal tracks can lead to longer routing paths, increased signal delays, and potential difficulties in completing necessary connections.

  - Timing Violations: Congestion can cause timing violations, where the desired timing constraints of the design cannot be met. When routing resources are heavily congested, signals may struggle to reach their destinations within the required time, resulting in timing failures and

potential performance degradation. This can lead to functional errors and compromised system performance.

- Power Delivery: Congestion can also impact the efficient distribution of power across the chip. Insufficient routing resources may hinder the proper placement and routing of power supply lines, resulting in increased resistance and voltage drops. This can lead to power supply noise, reduced performance, and potential functionality issues. Inadequate power delivery can also lead to reliability concerns and impact the overall power consumption of the chip.

- Signal Integrity Issues: High congestion levels can exacerbate signal integrity problems. When routing resources are concentrated in a small area, closely spaced routing tracks can induce unwanted signal interactions such as crosstalk and noise coupling. This can lead to signal distortion, increased noise levels, and compromised signal quality, potentially causing errors .

3.2.1 Option 1: Automatic floorplan:

3.2.1.Explain how the parameters core_utilization and side_ratio in set_auto_floorplan_constraints procedure effect the automatic floorplan.

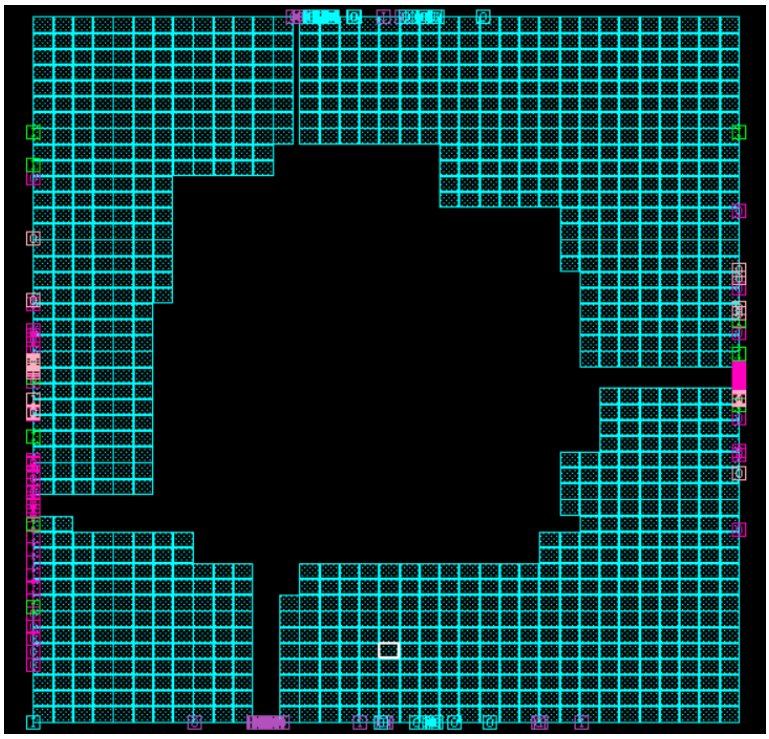
a. core_utilization: In the line set_auto_floorplan_constraints, the parameter -core_utilization is set to 0.5. This value represents the desired percentage of the chip's core area that should be utilized for placing the design blocks. A value of 0.5 means that the floorplan will aim to utilize 50% of the chip's core area for block placement. Adjusting this parameter will impact the density of the blocks within the available chip area. Higher values will result in a more compact floorplan, while lower values will leave more unused space within the core.

b. side_ratio: In the same line, the parameter -side_ratio is set to {1 2}. This parameter defines the aspect ratio of the chip's boundary or core area. Here, it indicates a ratio of 1:2 between the width and height dimensions of the chip. The side_ratio parameter can be adjusted to meet specific requirements or constraints of the chip design. It impacts the shape and dimensions of the core area, influencing the floorplan layout.

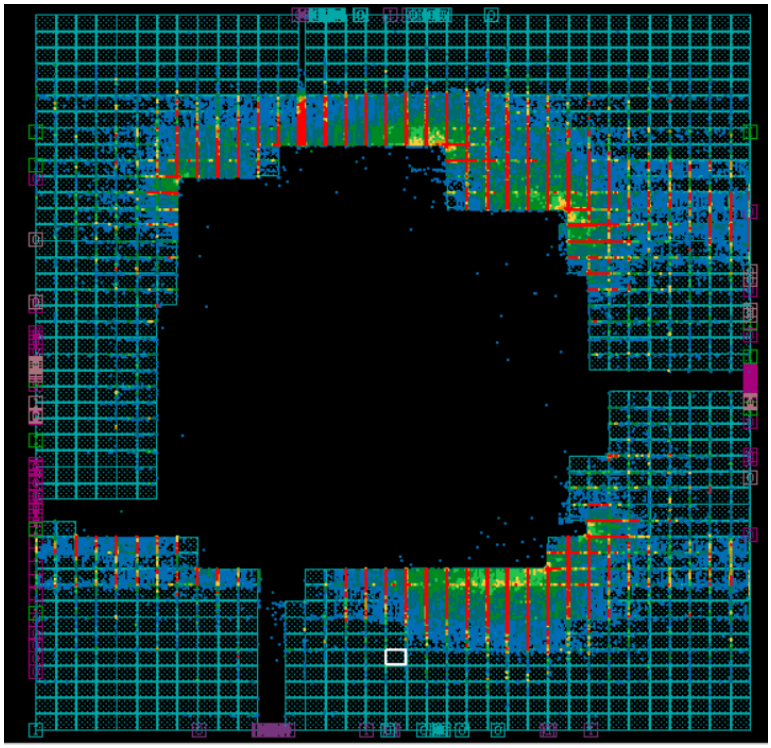
3.2.2.What metals are we using for the power grid?

metals M1, M5, M6, M7, M8, and M9 are used for different aspects of the power grid.

Screenshot without the congestion map :



Screenshot with the congestion map :



2.2 List the major pros and cons of the floorplan option.

Pros

- There is a well separated area for standard cells and macros(SRAMs). Stadard cells are present in central region and macros are present at boundaries of the design
- kind of design can lead to less congestion as there are very less blockages and a lot of space for routing from a standard cell to other standard cells. This basically gives large routing channels.

Cons of the floorplan option:

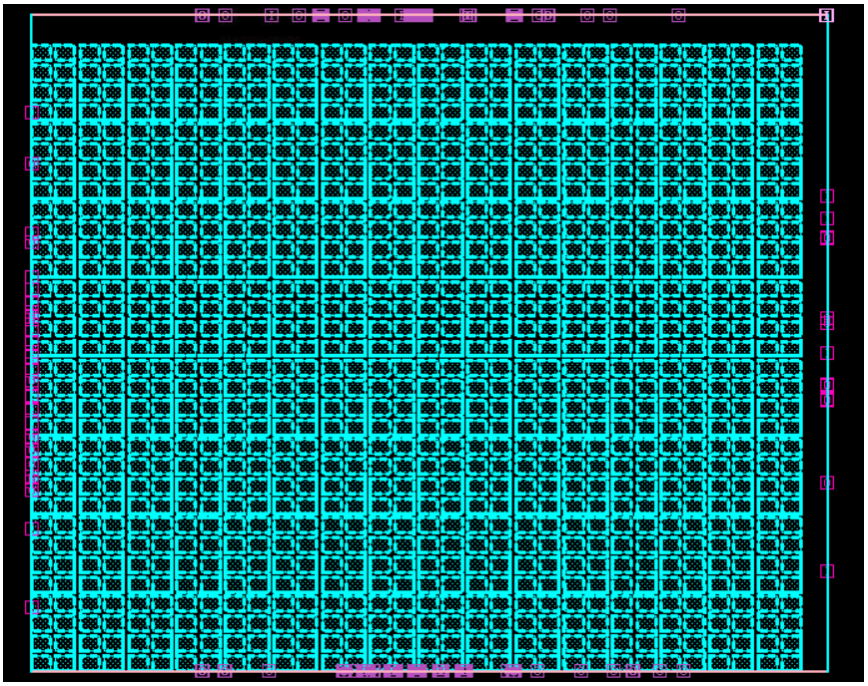
- There is a large density of SRAMs at boundaries. There is too less space between macros as compared to option 2. This is leading to large number of overflows at many points as can be seen in the congestion map.
- Because of high density of macros at corners there will be many pins in the narrow channel and near the corner. This can lead to large congestion.

2.3.Suggest how can the floorplan be improved (automatically or manually) and discuss the tradeoffs.

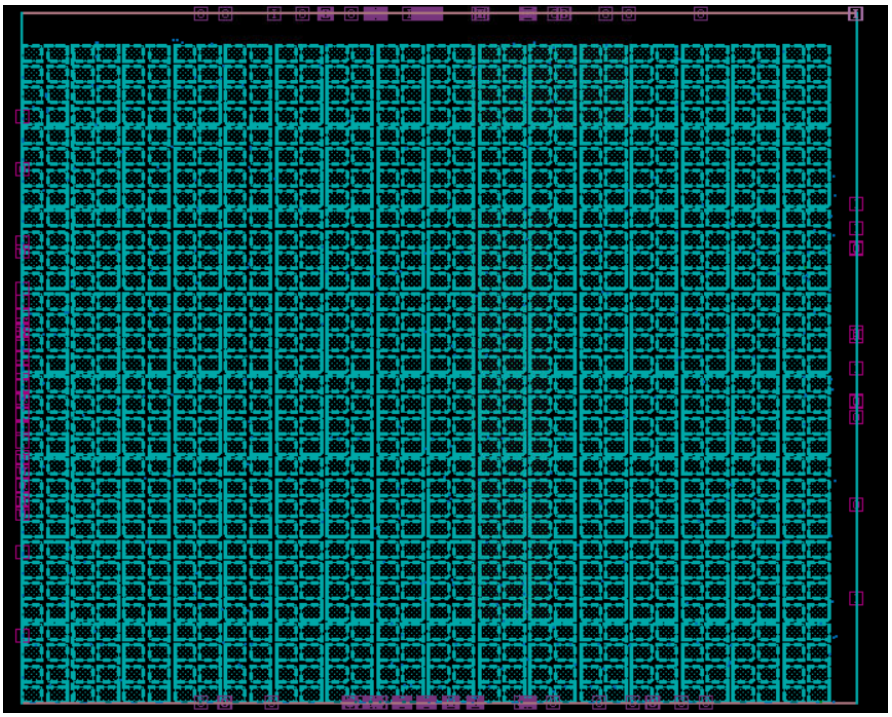
It can be improved by increasing the core utilisation. It's only 50% currently and can be improved further. By observing the layout closely we find that a lot of space at the central region is completely empty. It can be filled with standard cells and so there will be more space for the macros and they can be placed more evenly.

Option 2: Re-organized the location of the SRAMs into a matrix shape (fixed spaces) :

Screenshot without the congestion map :



Screenshot with the congestion map :



2.2 List the major pros and cons of the floorplan option.

Pros:

- There are no large overflows in the current design.
- The macros are more evenly distributed so more routing channels available between them as compared to previous design.

Cons:

- There is no well separated region assigned for standard cells and macros. This creates difficulty in routing between standard cells and macros as many macros come in the path and this creates a lot of congestion in the narrow paths available for routing.

2.3. Suggest how can the floorplan be improved (automatically or manually) and discuss the tradeoffs.

There should be a well defined area just for standard cells at the center of the layout and the area surrounding it can be covered with macros. This finishes any chances of bottlenecks and leads to less congestion. A proper utilization of blockages and appropriate spacing between macros will improve pin accessibility.

Option 3: Re-organized the location of the SRAMs into a matrix shape (unfixed spaces)

2.1. What does the function do?

The create_macro_arrays function has the main role the deciding the floorplan. The numbers we enter as parameters in it decide how the macros will be arranged. What will be the spacing between them. The number of columns and rows presenting the final layout.

2.2. How can we set the input arguments to get a floorplan like option 2?

To set the input arguments for getting a floorplan similar to option 2, we need to modify the create_macro_arrays procedure call. The input arguments for create_macro_arrays determine the number of rows, number of columns, width, and height of the macro arrays.

In the first call to create_macro_arrays, the arguments are:

8: Number of macros per array

4: Local number of rows

2: Local number of columns

10: Local width

20: Local height

32: Global number of rows

4: Global number of columns

50: Global width

50: Global height

To achieve a floorplan like option 2, we tried different combinations of these input arguments and we find that :

10: Number of macros per array

5: Local number of rows

2: Local number of columns

10: Local width

20: Local height

10: Global number of rows

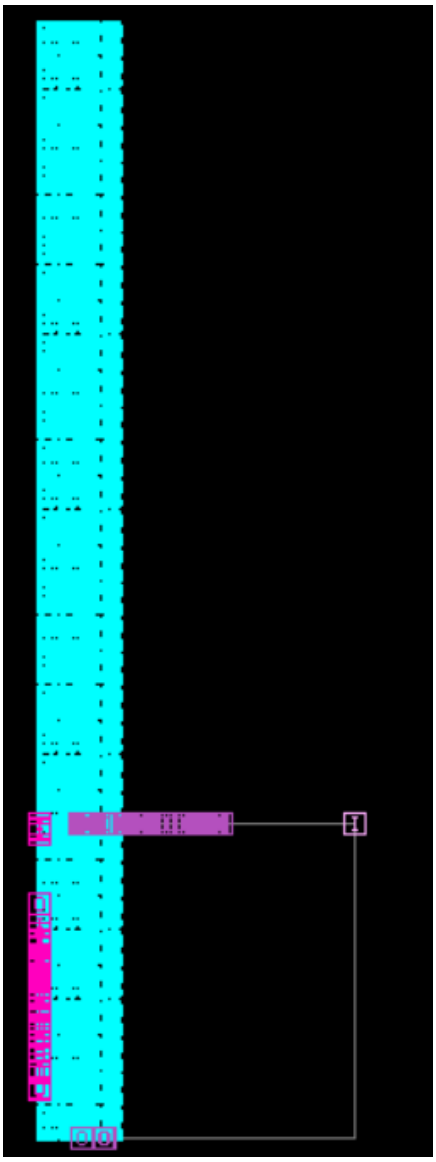
15: Global number of columns

50: Global width

50: Global height

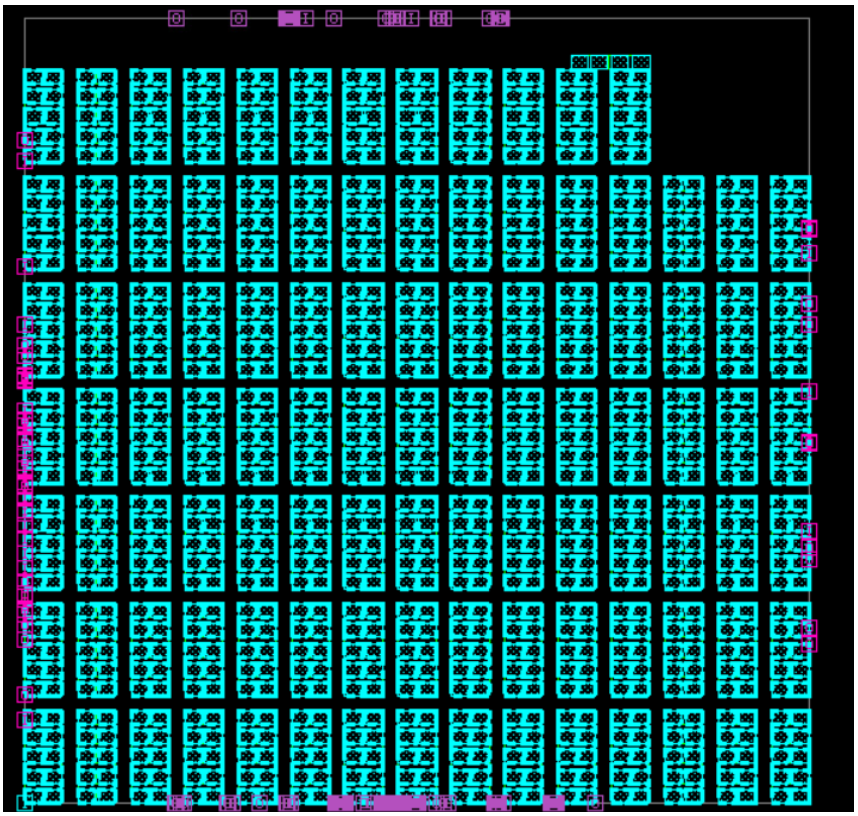
Task X :

Before changing the input we got :

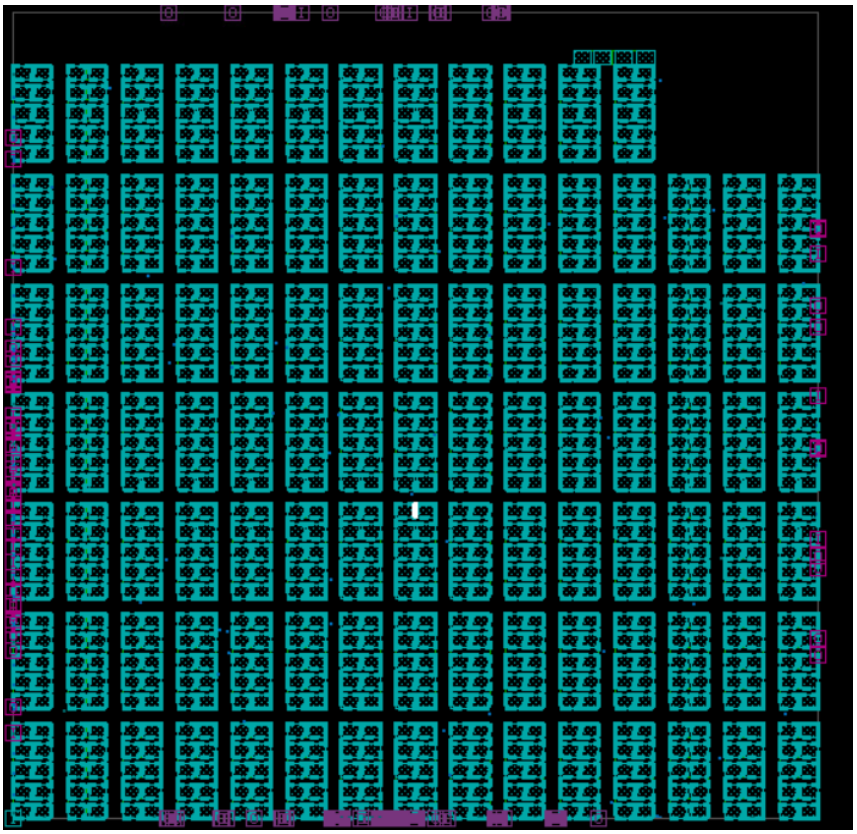


.

And after changing the input parameters :



Screenshot with the congestion map :



2.2 List the major pros and cons of the floorplan option.

Pros:

- The large space between each macro can be used to place large quantities of standard cells. So there will be well separated areas for standard cells and macros. This can help in reducing congestion. Also there is a relatively large space given between subsequent macros increases the amount of routing resources .

Cons:

- Due to the matrix type design there is still no dedicated area for placing the standard cells. It's always better to have a dedicated area for placing standard cells where there are no macros as obstacles. This reduces routing congestion and bottlenecks

2.3. Suggest how can the floorplan be improved (automatically or manually) and discuss the tradeoffs.

We can improve the design by creating a big region dedicated only for placing standard cells and rest of the region for placing macros. As stated earlier, this design reduces congestions and bottlenecks. We need to ensure that there is stll sufficient space between macros so that they are easily accessible to other macros and standard cells.

Option 4: Divide and conquer :

In this theoretical option for the floorplan, the bitcoin design is divided into 16 instances of a 4-bit top module, arranged in a 4x4 fashion. The 4-bit top instances in each row are abutted together. The second row is created by flipping the first row along the x-axis and then abutting it with the first row. The same process is followed for the third and fourth rows. Additionally, space is left in the middle for the placement of top-level cells.

2.2 List the major pros and cons of the floorplan option.

Pros

- This is an example of secondary hierarchy. The bit_coin level consists of 16 bit_top levels. So we just need to make sure that our bit_top level is build fine. Rest we just need to multiply it and abut the rows to build the bit coin stage. This technique increases the accuracy substantially.

Cons

- This technique can only be applied where secondary hierarchy exists. So this technique can be applied in very limited conditions.

[#P3.2_Q2]

a combination of Option 3 (Re-organized the location of the SRAMs into a matrix shape with unfixed spaces) and Option 4 (Divide and conquer) can indeed be a viable approach for improving the floorplan.

By dividing the design into smaller units (such as the 16 bit_top instances in a 4x4 fashion as described in Option 4), it reduces the complexity of dealing with a large number of macros. This division allows for better control over the placement and facilitates easier optimization of individual units.

Combining these two approaches can provide several benefits:

- Simplified Placement: Dividing the design into smaller units makes it easier to handle and optimize each unit's placement. It allows for better control over congestion, timing, and power optimization within each unit.

- Improved Area Utilization: The matrix shape organization, as in Option 3, can enhance the packing density of the SRAMs and maximize the usage of the available area. This can lead to better overall area utilization for the entire design.

- Reduced Complexity: Dealing with a smaller number of macros at a time simplifies the floorplanning process. It can improve routability, reduce congestion, and enable easier analysis and optimization of individual units.

[#P3.1_Q4] In what situations might a worker need to do manual work to assist an automatic program in floorplanning?

- Design Constraints: Floorplanning algorithms operate based on specified constraints and guidelines. In complex designs, there may be specific requirements that cannot be adequately captured by the automated program alone. For instance, certain macros or modules might need to be placed close to each other due to communication or power requirements. In such cases, a worker may need to manually intervene to enforce these constraints and guide the floorplanning process.

- Customization and Optimization: Automated floorplanning algorithms typically optimize for general objectives such as area, wirelength, or timing. However, there may be design-specific considerations or performance tradeoffs that require manual intervention. For example, if minimizing congestion in a critical circuit area is a priority, a worker may manually rearrange macros to alleviate congestion hotspots.

- Design Debugging and Optimization: Floorplanning is an iterative process, and sometimes the automatic program may not produce the desired results or fail to meet specific requirements. In such cases, a worker may need to manually analyze the floorplan, identify issues or bottlenecks, and make targeted modifications to improve the layout. This can involve adjusting the relative positions of macros, resizing modules, or redistributing critical paths.