VLSI Design for Manufacturability

Timing-Aware Dummy Fill Insertion

Name: 小山翼 (Tsubasa Koyama)
Student ID: 110062526

Ⅰ. How to compile and execute the program.

- To compile the program, navigate to the top directory and run the following command. This will generate an executable file named “Fill_Insertion”.

Command: $ make

- The program can be executed using the following command.

Command: $ ./Fill_Insertion <input file> <output file>

e.g.: $ ./Fill_Insertion ./input/testcase1.txt ./output/testcase1.txt

Ⅱ. Overall Flow

In this work, the timing-aware dummy fill insertion is proposed. Fig.1 provides an overview of the entire flow, which solves the dummy fill insertion on a layer-by-layer basis. For each layer, the corresponding layout is first converted into grids. Next, dummy fills are inserted into the area which is not affecting the lateral capacitance of the critical nets. Subsequently, a density refinement step is performed by inserting dummy fills near critical while ensuring minimal impact on the lateral capacitance of critical nets. The algorithms for ***Layout Grid Construction***, ***Initial Dummy Fill Insertion***, and ***Density Refinement*** are discussed in detail in Sections III, IV, and V, respectively.

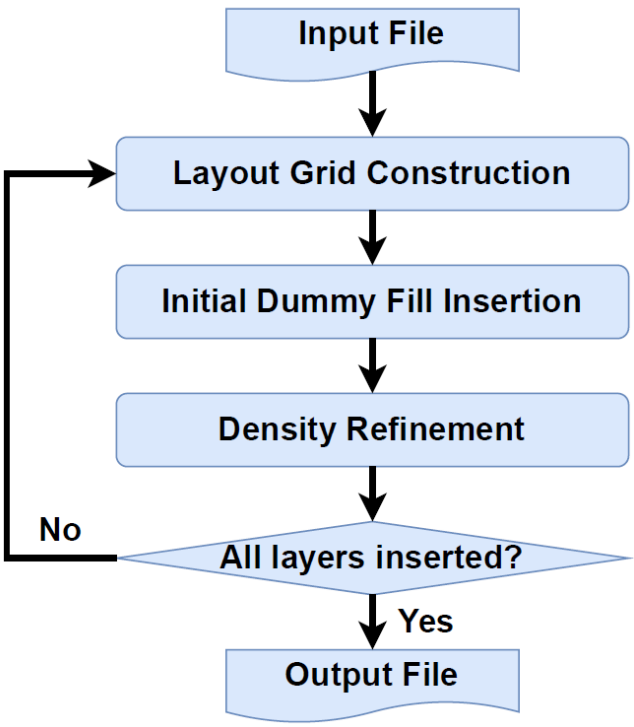


Fig.1: Overall Flow

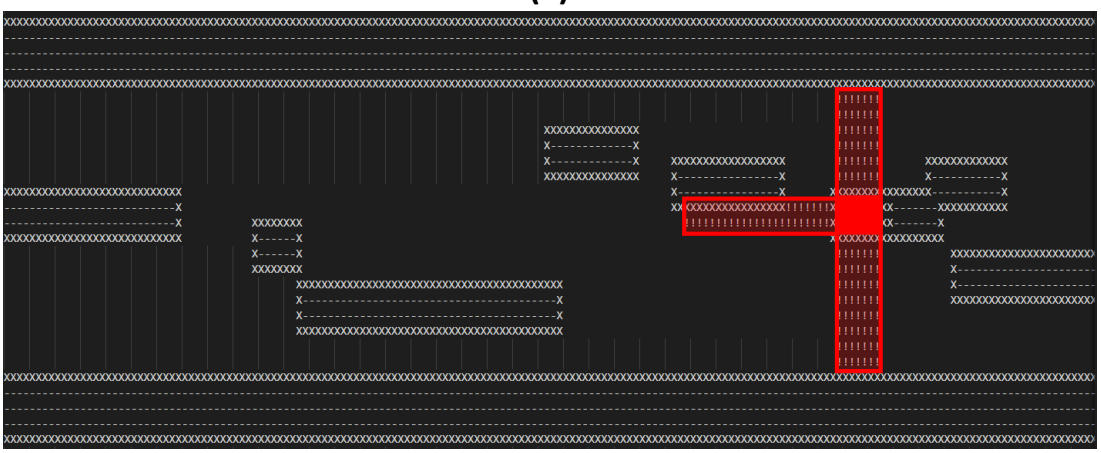
Ⅲ. Layout Grid Construction

The layout is initially converted into a 2D grid structure with a size of $row*col$, where each grid has a size of $grid\_size$. The values of $grid\_size$, row, and col are defined as follows:

$$grid\_size = max(minimum\ spacing, minimum\ fill\ width)$$
$$row = chip\ height / grid\_size$$
$$col = chip\ width / grid\_size$$

If the grid size is determined as the maximum value between the minimum spacing and the minimum fill width, the subsequent dummy fill insertion process does not need to consider spacing or fill width violations. During the layout conversion, the density of each grid is calculated to facilitate the subsequent density calculation. For each conductor, the adjacent grids will be designated as minimum spacing areas. Additionally, when inserting the conductor for the critical net, a keep-out area with a distance of 1600 is defined around it, where no dummy fill will be inserted in the initial stage. The routing direction of the layer will also be determined based on the combined height and width of the conductors. Fig.2 provides examples illustrating the grid construction process, with “-”, “X”, and “!” represents the conductor, minimum spacing area, and the keep-out area, respectively.

(a)



(b)

Fig.2: (a) The original layout with critical conductor and keep-out area represent as red solid rectangle and red translucent rectangle, respectively. (b) The converted layout grids.

IV. Initial Dummy Fill Insertion

After the ***Layout Grid Construction*** stage, dummy fills will be inserted into the available area, excluding the keep-out area. The insertion process begins from the bottom-left of the chip and proceeds strip by strip, following the routing direction, until reaching the top-right of the chip. To determine the maximum length of each dummy fill in the routing direction, a search is performed. Subsequently, another search is conducted to determine the maximum length perpendicular to the routing direction while maintaining the previously calculated length. Once the maximum area for the dummy fill is identified, it is inserted, and the process continues with the next dummy fill.

Fig.3 illustrates an example of the initial dummy fill insertion in a row-based routing layer with the dummy fills represent as green rectangles. Suppose that the current inserting dummy fill is i, the process begins by searching for the length in the routing direction (indicated by the red arrow). This determines the maximum length along the row. Following this, a search is performed perpendicular to the routing direction (indicated by the orange arrow), while maintaining the previously found length. Once the suitable area for inserting the current dummy fill (i) is identified, the insertion takes place, and the process continues with the next dummy fill (i+1).

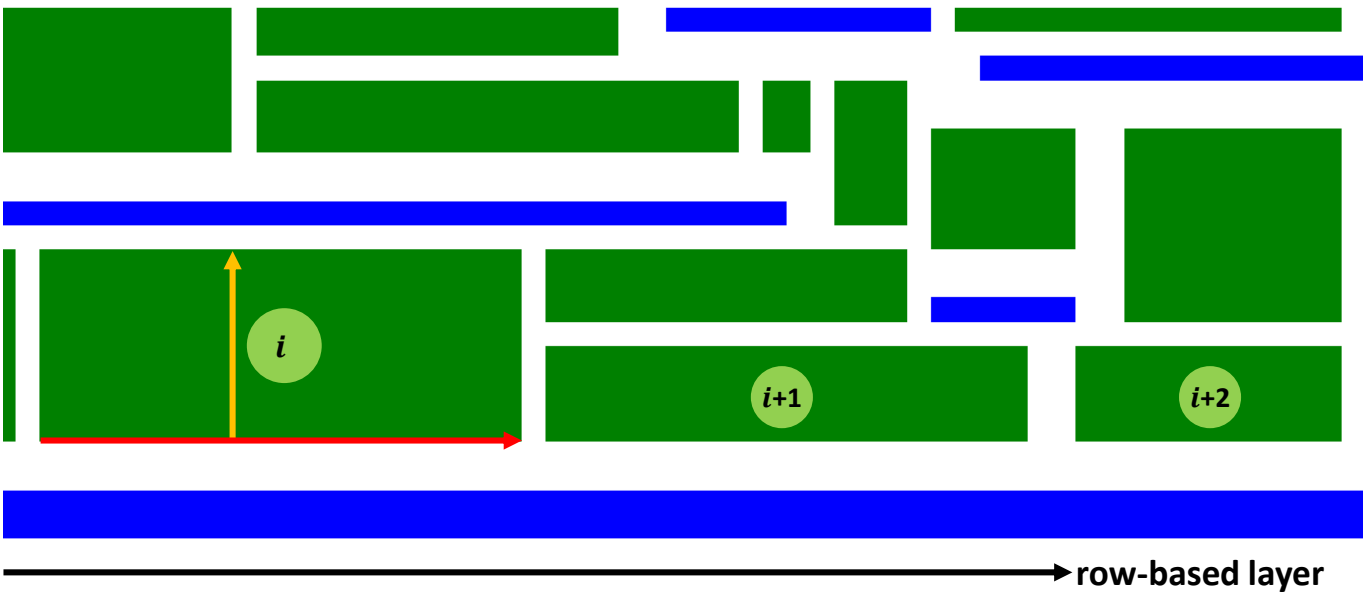
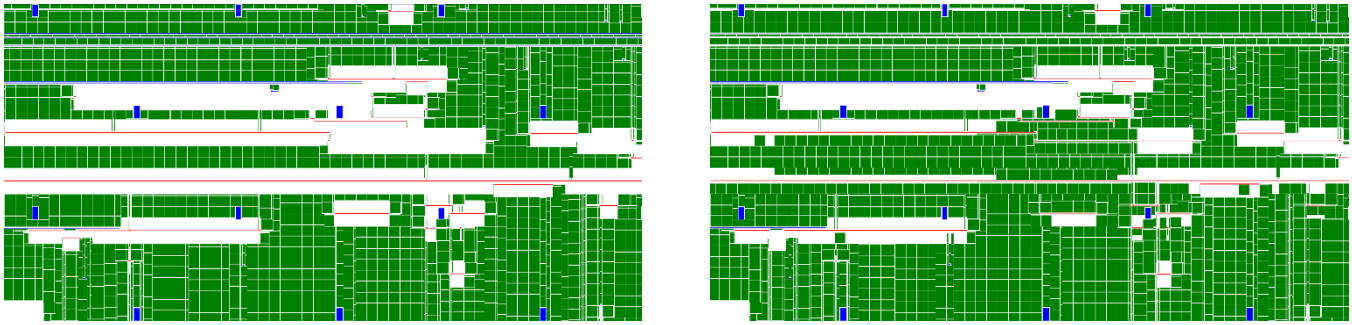


Fig.3: Illustration of the initial dummy fill insertion with the conductor and dummy fill represent as blue and green rectangles, respectively.

Ⅴ. Density Refinement

During the density refinement stage, the window density is calculated iteratively from the bottom-left to the top-right of the chip. For each window, the density is evaluated. If the calculated density falls below the minimum metal density requirement or exceeds the maximum metal density threshold, appropriate actions are taken. If the window density is lower than the minimum metal density, dummy fills are inserted into the keep-out area within the window. Conversely, if the window density surpasses the maximum metal density, existing dummy fills within the window are removed. These iterative calculations and adjustments are performed for each window in the chip, ensuring that the overall density meets the desired specifications by inserting or removing dummy fills as necessary. Fig.4 depicts an example of the density refinement process. From the comparison between (a) and (b), it can be observed that dummy fills are strategically inserted around the critical conductors to satisfy the metal density constraints.



(a) (b)

Fig.4: Illustration of density refinement stage.
(a) Before density refinement. (b) After density refinement.

VI. Experimental Results

Benchmark

Case	Chip Size	# Layers	# Conductors	# Critical Paths	Window Size
3	270000 * 170000	9	64903	55	10000
4	420000 * 210000	9	149464	100	10000
5	480000 * 350000	9	275425	171	10000

Experimental Results

Case	# Dummy Fills at Layer (L) 1~9									Total Weighted Capacitance	Runtime (s)
	L1	L2	L3	L4	L5	L6	L7	L8	L9		
3	59811	54275	37570	31063	27962	28541	21000	19446	3160	315614.00	174.66
4	133208	104915	77232	62873	54566	55931	40341	37289	5411	621691.70	42.22
5	253550	199524	143787	117683	101647	106317	76844	70018	10640	1266921.78	90.15

In the experiments conducted, all the constraints were successfully cleared. However, it was observed that the division process in the proposed grid-based method is not very flexible, resulting in longer runtimes for larger grid sizes, as seen in case 3. To address this issue and improve runtime efficiency, parallel programming techniques, specifically parallelogram programming, were applied. This approach involves distributing each layer to a unique thread, thereby reducing the overall runtime of the algorithm.

VII. Conclusions

In this work, a grid-based approach is employed for the insertion of dummy fill. The experimental results indicate that all constraints for each layer in all cases are successfully met. However, it is observed that the runtime can become time-consuming for larger grid sizes. As an alternative solution, the use of Hanan's method for grid construction is proposed as a potential approach to address the challenges encountered in the proposed grid-based method and further enhance its performance.