Metal Layer Pattern Detection

A Project report

Submitted in Partial Fulfillment of the Requirements for the Degree of

MASTER OF TECHNOLOGY
IN
ELECTRONICS & COMMUNICATION ENGINEERING
(VLSI DESIGN)

By

Tithi Bhavsar (18MECV02)

Under the guidance of

External Guide
Mr. Toney Manakalathil

Internal Guide
Dr. N.M. Devashrayee

Department of Electronics & Communication Engineering
Institute of Technology
NIRMA UNIVERSITY
Ahmedabad 382 481

May 2020

Certificate

This is to certify that the Internship Report entitled **"Metal Layer Pattern Detection"** submitted by **Ms. Tithi Bhavsar (18MECV02)** towards the partial fulfilment of the requirements for the award of degree in Master of Technology in the field of Electronics & Communication Engineering (VLSI Design) of Nirma University is the record of work carried out by her under our supervision and guidance. The work submitted has in our opinion reached a level required for being accepted for examination. The results embodied in this internship work to the best of our knowledge have not been submitted to any other University or Institution for award of any degree or diploma.
Date: April 28, 2020

Institute Guide
Dr. N. M. Devashrayee
Professor

PG Coordinator
Dr. Usha Mehta
Department of Electronics &
Communication Engineering
Institute of Technology
Nirma University
Ahmedabad

Director
Institute of Technology
Nirma University
Ahmedabad

Head of Department
Department of Electronics &
Communication Engineering
Institute of Technology
Nirma University
Ahmedabad

EXTERNAL CERTIFICATE

This is to certify that the Major Project Report entitled **"Metal Layer Pattern Detection"** submitted by **Tithi Bhavsar** (Roll No. **18MECV02**) as the partial fulfillment of the requirements for the award of the degree of Master of Technology in VLSI Design, Electronics & Communication Engineering, Institute of Technology, Nirma University is the record of work carried out by her under my supervision and guidance. The work submitted in our opinion has reached a level required for being accepted for the examination.

Date: 8/5/2020

Place: Bangalore

Toney Manakalathil
Engineering Manager
Intel Technology India Pvt Ltd
Bangalore

Acknowledgments

No task can be accomplished without the guidance, support and appraisal. I would like to heartily thank all those who helped me directly or indirectly in taking up and completing the project. I would like to express my sincere gratitude to Professor Dr. N.M Devashrayee for constantly guiding me throughout the internship and teaching me invaluable lessons, technically as well as morally.

I am grateful to Professor Dr. Usha Mehta, PG-Coordinator (M. Tech. VLSI Design) for continuous guidance and support throughout Internship project work.

I am also thankful to my external guide Mr. Toney Manakalathil, Engineering Manager, Intel Corporation, for providing good work and guidance during the internship.

Many thanks to the faculty members of Electronics and Communication Department for making me technically capable and skilled to take up the project.I am also grateful to Institute of Technology, Nirma University for the laboratory facilities without which the work wouldn't have been possible.

Lastly, I would also like to thank all those people who helped me in this project and the Almighty for his constant support throughout the project.

- Tithi Bhavsar (18MECV02)

Abstract

Chemical Mechanical Planarization (CMP) is a polishing process that does chip planarization to remove excess surface materials using chemicals. However, post CMP local and global planarization of the chip depends on the layout pattern density. To ensure that metal density is evenly distributed on the entire chip, insertion of dummy metal fill in the less dense areas of the design is important. To automate this process of dummy metal fill, the tool initially needs to detect all the existing metal layers of the design in order to avoid unwanted shorts with the signal lines and other DRC violations. This makes Metal Layer Pattern Detection an important stage to make sure that no violations occur post dummy metal fill. In this paper discusses about Pattern Detection Engine (PDE) which is used to determine metal tracks in the design.

Pattern Detection engine is the flow that detects metal patterns existing in the design and based on it decides which metal pattern can be placed in the white space region in between the existing metal patterns. These patterns are determined using Constraint Programming of CPLEX optimizer which gives an optimal solution in terms of transition patterns i.e. the pattern that can be placed between the existing metal track patterns.

Apart from the PDE flow, this paper also discusses various new methods developed to improve the metal ll results in terms of both DRC's and Density violation. Also, there were new techniques like bounding box and incremental fill enabled to help users in the ECO stage, prior to final sign-off of the design.

The new PDE flow developed could solve minimum density violation efciently. Also, many DRC violations could be solved with new PDE ow as it added metal patterns as per the metal rule book. Bounding box and Incremental fill techniques reduced the manual efforts that a designer might need to take before running fill and thus helped them to reach sign off stage for their design faster.

Contents

Acknowledgements **iii**

Abstract **iv**

List of Figures **vii**

1 Introduction **1**
1.1 Fill in VLSI Design 1
1.2 Need of Metal Fill Placement 3
1.2.1 Pattern Detection Engine 4

2 Literature Survey **5**
2.1 Chemical Mechanical Polishing in VLSI 5
2.2 Need of Fill for better CMP 6
2.2.1 CMP Fill Benefits and Trade Offs 7
2.3 Pattern-Dependent Variation in CMP 8
2.3.1 Effect of metal fill on CMP 8
2.4 Pattern Detection Engine (PDE) flow 9
2.5 CPLEX optimizer 10
2.5.1 Constraint Programming in CPLEX optimizer 10

3 Implementation **12**
3.1 Implementation of Bounding-Box utility 12
3.2 Incremental Fill 13
3.3 Use of CP to implement constraints for metal patterns 15
3.4 Pattern Generation for PDE Evaluation 16

4 Results **19**
4.1 Fill using Bounding Box 19
4.2 Incremental Fill to reduce manual efforts 21
4.3 Result of implementing new PDE Flow 23

5 Conclusion and Future Scope **25**

References **26**

List of Figures

1.1 Growth of number of components per MOS IC chip 2
1.2 Dishing and erosion of metal layer due to non-uniform layout density 3
1.3 Effect of fill on CMP 4

2.1 CMP process steps for base layers and metal layers 6
2.2 Effect of dummy metal fill 7
2.3 PDE flow 10

3.1 Performing metal fill only in specified region 13
3.2 Container names when fill run for first time 14
3.3 Container names when fill run for second time 15
3.4 End-to-end metal tracks in the input layout design 17
3.5 Random metal tracks in the input layout design 17
3.6 Single arrangement metal pattern generated by the script 18
3.7 Dual arrangement metal pattern generated by the script 18

4.1 Rectangular metal fill done using bbox command 20
4.2 Triangular metal fill done using bbox command 21
4.3 Second instance of metal fill done in absence of incremental fill . . 22
4.4 Second instance of metal fill done with incremental fill enabled . . 23
4.5 Runtime degradation in metal fill stage 24

Chapter 1

Introduction

1.1 Fill in VLSI Design

Chemical-Mechanical Planarization (CMP), referring to the topographical planarization of the dielectric layers, is a very important step in the chip manufacturing process. Density of the layout pattern is a major factor affecting post CMP local and global planarization. If the layout patterns are not uniform, then it affects the polishing of surfaces by CMP which can be due to metal dishing and dielectric erosion. This eventually results in a worsened lithography output. This causes printed layout patterns to be out-of-focus which ultimately affects the performance and yield of the layout design. This can also cause incorrect patterns getting transffered to the wafer during lithography. Design for manufacturability also plays a vital role in such situations.

Ultra-large-scale integrated (ULSI) circuits are now being fabricated on semiconductor substrates, or wafers. This has reduced the cost of manufacturing and increased their efficiency. Below is the figure that shows how the number of components on a MOS memory chip has grown over years.

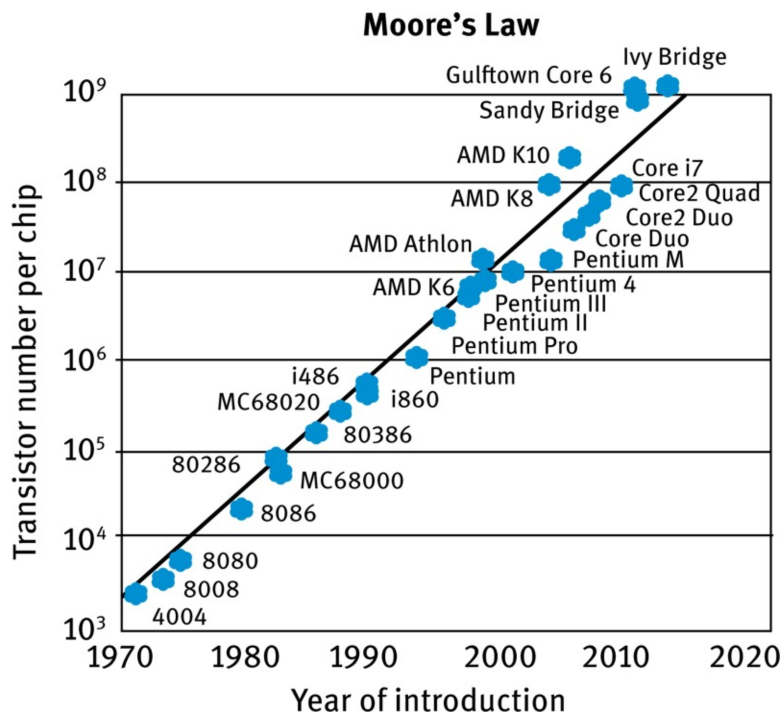

Figure 1.1: Growth of number of components per MOS IC chip

As CMOS technology advances in keeping up with the Semiconductor Industry Association National Technology Roadmap for Semiconductors and moves into the 180-nm generation and further with reducing technolgy, design needs to be driven majorly keeping in mind the manufacturing cost. The action or process of gradually writing off the initial cost of a foundry, now becomes a dominant business concern. If the patterns are not uniform in the layout it can result into non-uniform polished surfaces resulting from metal dishing or dielectric erosion as shown in Figure 1. This ultimately leads to a bad lithographic output.

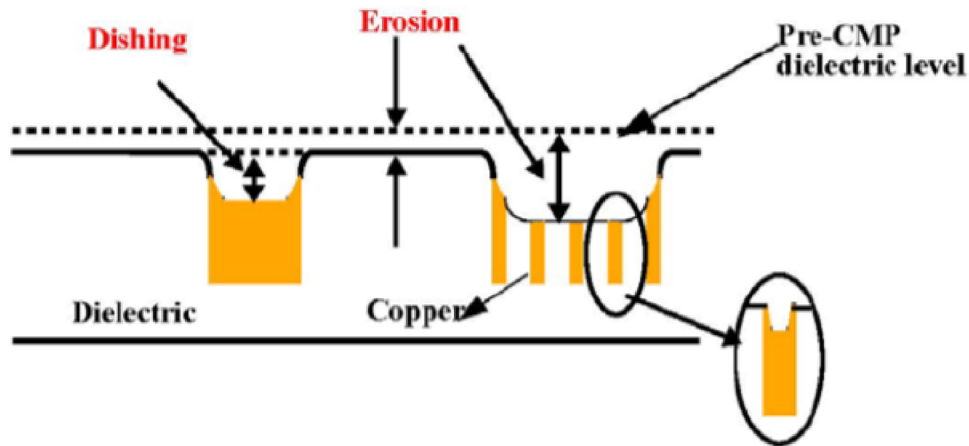

Figure 1.2: Dishing and erosion of metal layer due to non-uniform layout density

In such scenarios manufactured devices must be predictable and uniform so as to maximize yield.Also process engineers must look into other interconnect and base layers related attributes like dopant concentrations, channel lengths, interconnect dimensions, contact shapes and parasitics, and interlayer dielectric thicknesses to maintain uniformity. This situation asks for development and auto action of fill flows. Currently we are working on an agenda to develop a flow in which a user only needs to run the fill flow and forget about all the design rule check (DRC) and density violations that might be occurring prior to fill being done.

1.2 Need of Metal Fill Placement

Propelled IC fabricating forms utilize Chemical Mechanical Polishing (CMP) to planarize metal and dielectric layers. CMP alone gives great local consistency, yet doesn't ensure global consistency. Be that as it may, CMP related to a metal thickness rule improves global consistency. The metal thickness rule requires including sham metal fill structures in regions of low metal density.As CMOS innovation downsizes, new materials are presented all the while, for example, copper for the metallization and low-K materials as interlayer dielectrics. To ensure the layer planarization in the CMP arrange, a uniform metal thickness is required in 90 nm or beneath forms for all metallization layers, ordinarily somewhere in the range of 30% and 90%. The metal thickness is checked by neighbouring "scan windows", which can be as little as 20×20 m $\sim$ i2for case, over the entire chip. Impact of fill on CMP can be seen in beneath figure. Fig a. shows non-uniform surface while in fig b. subsequent to doing sham fill CMP gives progressively

uniform surface.

Figure 1.3: Effect of fill on CMP

To meet this requirement, it becomes necessary to add dummy metal fill layers along with the existing components in the design.[1]

However, a dis-advantage of this can be an increase in the parasitic capacitance which can increase the mismatch in the design blocks. Metal fill degrades the quality factor of spiral inductors used in an oscillator circuit; furthermore, it impacts the power consumption and phase noise of the circuit. Furthermore, floating metal structures are commonly used as an inherent part of passive circuits.[2] This is one of the trade off of metal fill which needs to be taken care of.

1.2.1 Pattern Detection Engine

Pattern matching is best known for its use in detecting lithographic hotspots, but it's also widely used across all physical verification flows, and has expanded into design-for-manufacturing (DFM) flows as well. Integrating pattern-matching functionality into DFM operations ensures that designs are quickly and accurately optimized for reliability, performance, and manufacturing prior to tapeout. Combining pattern matching with DFM vector profiles has a wide range of uses in design enhancement.

Insertion of dummy metal fill is required for maintaining the local and global uniform density. However prior to doing the dummy metal fill, it is important to do detect the existing metal tracks in the design so that no unnecessary shorts or open are created while inserting the dummy metal fill. This makes development of Pattern Detection Engine an important step for metal fill. Main purpose of PDE is to correctly identify the incomplete metal tracks patters in different parts of the entire design. Once the regions with different metal patterns are identified, it needs to identify the transition metal fill pattern that can be placed between any two different metal pattern regions to satisfy lower and upper bound of metal density.

Chapter 2

Literature Survey

2.1 Chemical Mechanical Polishing in VLSI

Chemical Mechanical Planarization (CMP) is a polishing process, which utilizes a chemical slurry formulation and mechanical polishing process to remove unwanted conductive or dielectric materials on the silicon wafer, achieving a nearperfect at and smooth surface upon which layers of integrated circuitry are built.If the surface materials are to be removed it can be called polishing and when the surface is to be made even it can be called planarization. CMP originally came into existence since etching of via's was not possible through dry etching. This was because vias were made of copper and so its etching could not be done using dry etch. As a result CMP was used for via etching and now it is used for complete chip planarization and poilishing.

Chemicalmechanicalcleaning/planarization(CMP)wascreatedinthelate1980s so as to beat issues with multi-layer metallization. The expanding geology because of stacked metal lines prompted depth-of-focus issues during photolithography and to unwavering quality issues brought about by metal line diminishing. The successful planarization by CMP of the between level dielectric layers permitted the manufacture of in excess of three metal layers. Without CMP, logic devices of present day rationale gadgets with up to 12 metal layers couldn't be manufactured. In this way, CMP is one of the empowering advances of the present day devices.

Throughout the years, the advancement of different CMP forms for "More Moore" simplied the handling of logic devices and also permitted complex use of the logic, for instance, the acknowledgment of copper metallization by introducing damascene and dual-damascene technology. In today's sub-14nm logic device fabrication,the number of CMP steps required in front-end-of-line and integration

comes to up to 18–20 (Moon et al., 2014). For instance, the presentation of nFET innovation with Replacement Metal Gates (RMG) prompts the assignments of acing the basic CMP steps. The fundamental advances being followed in CMP are appeared in the underneath gure.

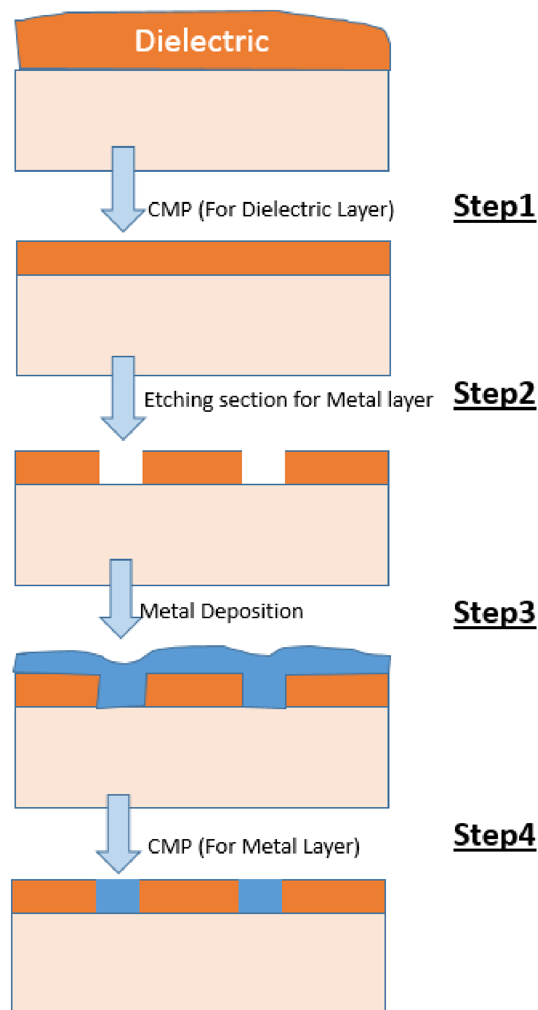

Figure 2.1: CMP process steps for base layers and metal layers

2.2 Need of Fill for better CMP

In the very deep-submicrometer, very large-scale integration-regime manufacturing steps, including optical exposure, resist development, and etch, and CMP have varying effects on device and interconnect features depending on local properties

of the layout. Foundry economics dictate that the process-window volumes be maximized, which in turn requires that device and interconnect features be fabricated as predictably and uniformly as possible. To achieve this goal, the layout must be made uniform with respect to a certain density parameter. The physics of semiconductor processing make predictable and uniform manufacturing difcult. In particular, the quality of post-CMP depends on the pattern density of the layer beneath a given dielectric layer.

2.2.1 CMP Fill Benefits and Trade Offs

Traditionally, foundry-supplied design rules have been used by the designers to meet density requirements while not significantly increasing the interconnect capacitance. While fill-insertion design rules have sufficed until now, they are overly conservative and arguably at the end of their life cycle. Rules have been used to limit the impact of fill on total and coupling capacitance. The figure below shows the effect of dummy floating metal fill on nearby interconnect.

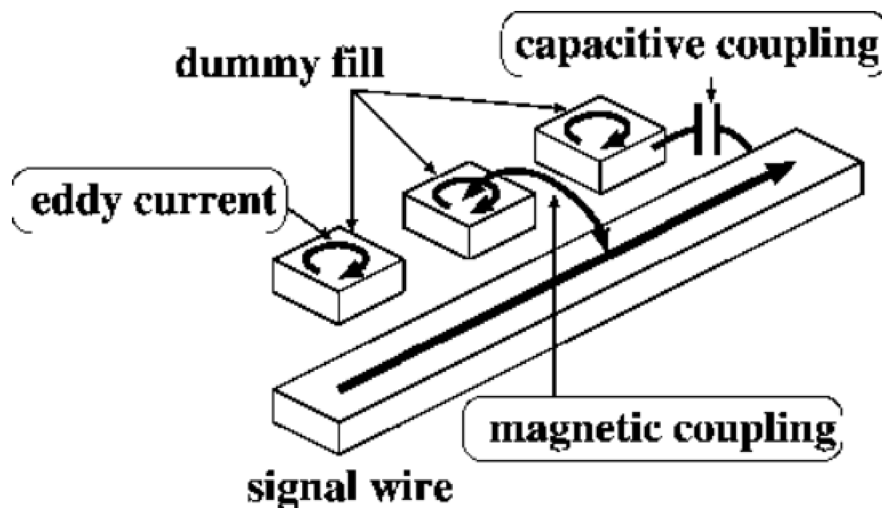

Figure 2.2: Effect of dummy metal fill

CMP fill insertion, even as it contributes to layout pattern-density uniformity, increases the coupling and total interconnect capacitance. This in turn, increases the uncertainty in circuit-timing calculations.

2.3 Pattern-Dependent Variation in CMP

CMP is very effective at reducing featurelevel or local step height and achieves a measure of global planarization not possible with spin-on and resist etch back techniques; however, CMP processes are hampered by pattern sensitivities which cause regions on a chip to have thicker dielectric layers than other regions due to differences in under lying topography. This problem has become especially acute as performance requirements have increased and dimensions have scaled. Also, CMP has found wider application in the entire VLSI development and production cycle serving as an enabling tool for shallow trench isolation and other novel process techniques.

2.3.1 Effect of metal fill on CMP

Since lithography can have aggressive effect on circuit performance and thus finally on the yield, Interlevel dielectric (ILD) variation must be kept in control. As the performance requirements increase with the new technologies coming up this problem has become bigger. Also, CMP has found wider application in VLSI technology development and production serving as an enabling tool for shallow trench isolation, damescene metallization technologies [8], and other novel process techniques.Attempts to control CMP intra level dielectric thickness variation include an exhaustive search for and experimentation with different consumable and process choices(especially pads),but no consumable choice currently available appears to reduce appreciably pattern-dependent dielectric thickness variation [9]; thus, the only viable choice available for reducing layout pattern dependent dielectric thickness variation is to change the layout pattern itself via the introduction of metal ll patterning. Metal-fill is a technique in which large white space regions in the design are filled with dummy metal wires which are left floating in order to reduce interlevel dielectric thickness.

However it is important to note that metal fill cannot just be done locally for an individual block, but it needs to be done during every step of integration of the topmost block. Also while doing metal fill other DRC violations and LVS issues must be kept in mind as it might lead to completely non-functional blocks. The dummy metal tracks added should not be very long or it might cause problems like electro static discharge.

The improvements in dielectric thickness uniformity observed in metal-ll experiments can ultimately beat tributed to layout pattern-density. The strong correlation between layout pattern density and dielectric thickness is well recognized for CMP

processes [4], and it has also been demonstrated that layout pattern-density is the primary variable controlling CMP-induced intradie dielectric thickness variation[4].

2.4 Pattern Detection Engine (PDE) flow

Pattern matching is divided into two stages: capture and match. In the capture stage, designers select areas of interest, using either graphical selection or pattern capture commands. Designers also have the flexibility to define patterns based on marker layers or hotspot geometries, and to define allowed variations, such as orientations, halo size, etc. Input to the pattern-matching process can be filtered by selecting locations and polygons with certain DFM properties.

Prior doing any metal fill in the input design layout, it is necessary to determine the existing metal patterns in the design. This metal patterns used by the designer are predefined by the DFM rules and these rules help in determining the existing patterns. However the metal tracks in the input design would never be complete with full end-to-end length and this makes the task of identifying patterns in the design difficult. To determine the metal pattern present runsets are developed which first learn the metal patterns and then try to follow those patterns on the design. If the pattern matches then the matching region is characterized to that pattern. The entire design is scanned in a similar way to characterize different regions into different metal patterns. Once all the existing patterns are determined, with the help of predetermined constraints list, transition metal patterns in the empty region between two metal patterns are determined. This process is done with the help of a solver, which takes constraints as an input and solves the equations based on these constraints to give an optimal solution. Here, an user input file (UIN) is also available so that user can add more constraints to the pre determined constraints list. Following are the steps followed for detecting metal patterns on the design layout:

- Scan input layout design for metal patterns
- List out constraints o determine which patterns can be placed in the intermediate empty regions
- Use a linear program solver to solve those constraints and obtain an optimal solution
- Enable (user input) UIN files to allow constraints from users

Following block diagram describes the flow of Pattern detection engine;

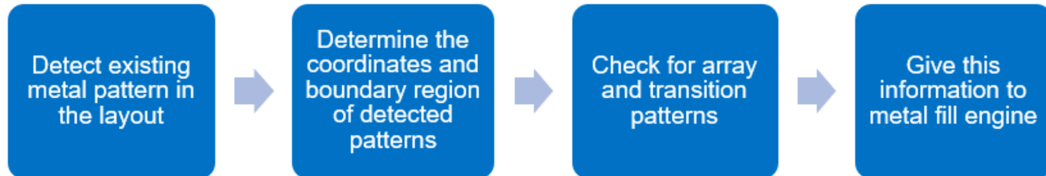

Figure 2.3: PDE flow

2.5 CPLEX optimizer

CPLEX Optimizer provides flexible, high-performance mathematical programming solvers for linear programming, mixed integer programming, quadratic programming and quadratically constrained programming problems. These solvers include a distributed parallel algorithm for mixed integer programming to leverage multiple computers to solve difficult problems. It is an executable program that can read a problem interactively or from files in certain standard formats, solve the problem, and deliver the solution interactively or into text files. The CPLEX Python API is a Python package named cplex that allows the Callable Library to be accessed from the Python programming language. It is equally suitable for interactive use through the Python interpreter or for writing scripts or full-fledged applications. IBM CP Optimizer is an important complement to the optimization specialists' toolbox for solving real-world operational planning and scheduling problems. CP Optimizer contains a robust optimizer that handles the side constraints that are invariably found in such challenges.

2.5.1 Constraint Programming in CPLEX optimizer

Solution to problems which can have combination of multiple optimized solution can be easily determined with the help of constraint programming technology. It is not based on mathematical linear algebra programming, instead it is developed on logic programming and graph theory which is based on computer science fundamentals. Constraint based programming is extremely helpful where there can be more then one solution to the problem but he user needs to identify the optimal solution based on the constraints provided by them. Here the complex logical relationship between the arithmetic decision variables makes use of this optimizer

inevitable. The solution given by constraint programming depends whether the objective has to be minimized or maximized.

Constraint programming works first to decrease the arrangement of potential estimations of the decision factors that fulfill all the imperatives by utilizing consistent, graphical, and different other contentions. At the point when the reasoning that a few values from the constraints provided are impractical, this data is engendered through the imperatives empowering further conclusions. Different searching algorithms are likewise utilized until a worth is doled out to each decision variable, that is, until an answer is found. After a first arrangement is discovered, the search algorithm continues to discover further arrangements with better target esteems. A significant advantage of utilizing CP Optimizer is that models can be figured and settled proficiently, whether or not the significant time scale to depict a scheduling issue is in milliseconds, minutes, or hours.

Constraint programming in context of Metal Pattern detection can be extremely useful to determine the transition patterns between already existing metal track patterns in the layout. On using maximize objective it can give us the optimal solution to decide on a pattern based on the constraints provided according to the metal rule-book.

Chapter 3

Implementation

3.1 Implementation of Bounding-Box utility

Metal fill insertion plays an important role to ensure uniformity of pattern-density necessary for better CMP. Also, metal fill in the design leads to lower layer-to-layer capacitance values than would be seen in a non metal-fill layout. This leads to requirement of being able to perform metal-fill in the areas which are either less densely populated or are having white space in the design. After the design is complete, layout designers need to fill the empty regions with metal fill. This metal fill can be accomplished using two methods:
1. Grounded metal fill
2. Floating metal fill
Both of this methods have their own pros and cons and is left to the designer to decide which would suit their requirements better.

For implementing bounding box utility a runset was developed bbox.rs. This runset is called whenever user uses the bbox switch with fill command line input. On using this bbox switch, a global keep out region would be created on the input layout with the coordinates provided by the user. All the base-fill layers and metal-fill layers would then be filled only in the bounding box specified by the coordinates provided by the user. The entire remaining region of the input design would be treated as a keep out region and no fill would take place their. This utility can be used to determine the pattern in a particular region of the input design. Below is the part of the runset developed to generate an intermediate global keep out region whenever bbox switch is enabled by the user.

```
BND = copy_by_cells(CELLBOUNDARY,{get_top_cell()}, CELL_LEVEL);
BND = flatten_by_cells(BND,{"*"});
p1 = coordinate_list;
```

```
note("DBG" + p1);
KOR = polygons(p1);
```

Following is the layout generated after doing metal fill for only bounding box.

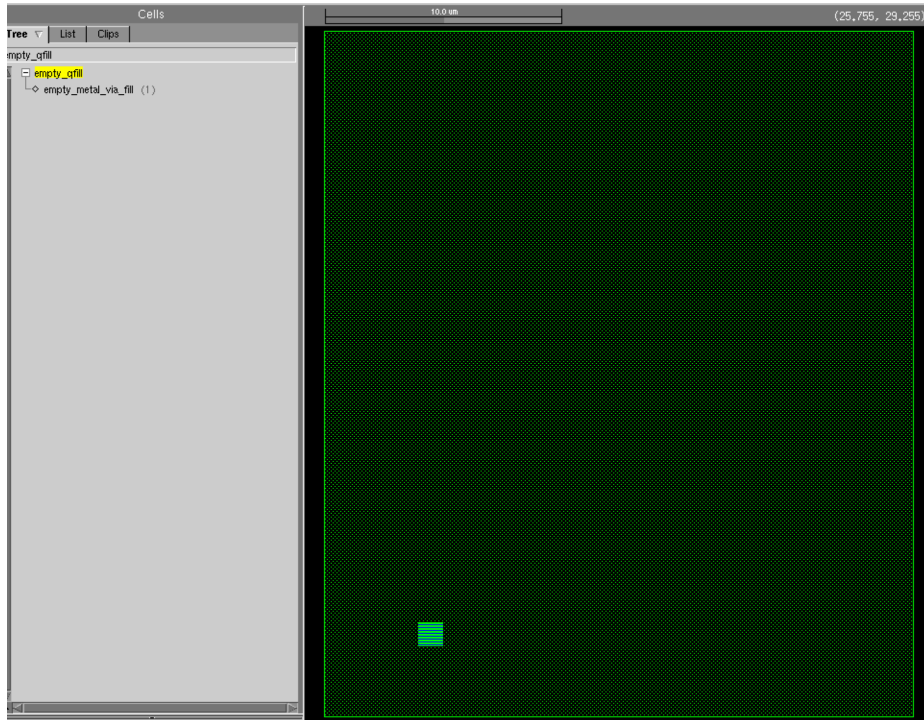

Figure 3.1: Performing metal fill only in specified region

3.2 Incremental Fill

Incremental fill will be helpful when user wants to do eco runs. The existing Fill API does not have any capability to replace existing fill containers generated on running fill for the first time. Suppose an user is running metal 2 fill on the input design on he entire design to meet meal 2 density. This would generate a fill container with name:
Top_cell_name_ metal_fill.
After running fill this container is then merged with input design to introduce this fill patterns in the design. Now imagine the user again wants to run metal 2 fill in a small area which still has minimum density violation. This can be done with the help of bounding box utility. So now on running metal fill for metal 2 layer in a specific region, a container with name top_cellname_metal_fill will again be created, which then has to be merged with input design. However the input

design already has container named top_cellname_metal_fill so when this new container is merged with input design the already existing metal fill container gets replaced by the new container. As a result the metal fill that was done earlier is removed from the design which would cause many more density violations. To avoid such scenarios user had to rename the already existing metal fill containers before merging the new fill container with input.

With development of Incremental fill user need not rename fill container names before running any ECOs. For each stage, we check if a container by name '<topcell>.*<container_suffix>' is already present in input layout. If it exists , fill UIN variable(output_container_suffix) will be modified to have "_timestamp" at the end.

As shown below, 1st image has the container names of all fill layers when QFILL is run for the first time.

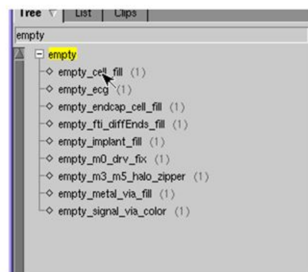

Figure 3.2: Container names when fill run for first time

The below image shows container names after second FILL run on output generated by first run as shown in above figure. In second run it finds a cell with name <topcell_name>_<stage_name> already existing in the input layout , so FILL adds " _timestamp" suffix to its container names as shown below.

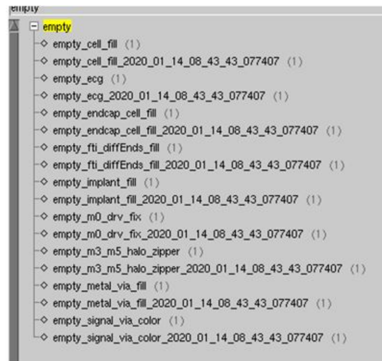

Figure 3.3: Container names when fill run for second time

3.3 Use of CP to implement constraints for metal patterns

Below is an example of CPLEX python program similar to the one written for metal layer pattern detection: The problem depicted here is similar to determing a transition pattern between existing patterns. The problem involves deciding colors of flags for six neighbouring states of India in a way that maximum four colors (blue, white, yellow, green) are used and no neighboring states have the same colored flag. Consider following six states: Maharashtra, Karnataka, Kerela, Tamil Nadu, Telangana, and Andhra Pradesh. Below is the script that gives me an optimal solution.

```
from docplex.cp.model import CpoModel
#Below line creates a CPO model
mdl = CpoModel()
mdl.print_information()
#Create model variables containing colors of the countries
Maharashtra = mdl.integer_var(0, 0, "Maharashtra")
Karnataka = mdl.integer_var(0, 1, "Karnataka")
Kerela = mdl.integer_var(0, 2, "Kerela")
Tamil Nadu = mdl.integer_var(0, 2, "Tamil Nadu")
Telangana = mdl.integer_var(0, 1, "Telangana")
Andhra Pradesh = mdl.integer_var(1, 3, "Andhra Pradesh")
ALL_STATES = (Maharashtra, Karnataka, Kerela, Tamil Nadu, Telangana, Andhra Pradesh)
#Add constraints to determine which countries can have same colored flag
mdl.add(Telangana != Maharashtra)
mdl.add(Kerela != Karnataka)
```

```
mdl.add(Tamil Nadu != Kerela)
mdl.add(Karnataka != Tamil Nadu)
mdl.add(Andhra Pradesh != Telangana)
mdl.add(Tamil Nadu != Andhra Pradesh)
mdl.add(Karnataka != Maharashtra)
mdl.maximize(1)
#Solve model
msol = mdl.solve(TimeLimit=1)
#print("msol is
if msol:
print("Solution status: " + msol.get_solve_status())
colors = ("Yellow", "Red", "Green", "Blue")
for state in ALL_STATES:
print(" " + state.get_name() + ": " + colors[msol[state]])
else:
print("No solution found")
```

The output of the above script is as below:

Solution status: Optimal
Telangana: Yellow
Tamil Nadu: Yellow
karnataka: Green
Maharashtra: Red
Kerela: Red
Andhra Pradesh: Red

So the output shows that if the constraints are as per above script then we can get an optimal solution i.e. the best solution for given constraints would be to have Telangana and Tamil Nadu as same yellow color, Maharashtra, Kerela and Andhra Pradesh can have red color and Karnataka can have green color. If the constraints here were provided in a different way solution could either be feasible or there could have been no solution. Thus deciding the constraints as per the metal rules is the most critical step in this flow.

3.4 Pattern Generation for PDE Evaluation

Once the pattern detection evaluation is enabled we need to validate that the patterns matched by PDE are correct or not. For this validation we need to generate test cases for pattern validation. Testing of PDEcan be done on two types of test cases.

- Full patterns These are the test cases with only end to end metal tracks. To validate such patterns following steps are run:

Run PDE on this test case
A GDS file based on the metal pattern detected by PDE generated
XOR test of the input GDS file and output GDS file
If the XOR test gives some pattern in the output then it shows that PDE failed

Figure 3.4: End-to-end metal tracks in the input layout design

- Random patterns This are the metal tracks that might be staggered in an actual design layout.To validate such patterns following steps are run:
  Direct XOR test here would not be efficient
  Determine a corresponding end to end metal track test case matching the staggered metal tracks
  Follow the Full pattern testing procedure on the determined test case

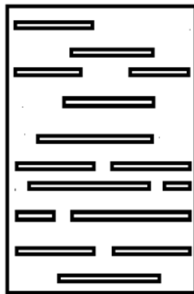

Figure 3.5: Random metal tracks in the input layout design

Manually layout of metal tracks were generated to test PDE earlier, but for rigorous and efficient testing of PDE more number of test cases have to be developed. To automate this process of test case generation a ruby script was developed that could generate all types of metal pattern test cases that could be found on the

input layout design. These test cases can be used to validate both full patterns and random patterns metal tracks. PDE would be run on these test cases and then to validate that PDE recognized the correct patterns, a XOR match test is done on the input of PDE and output of PDE. If the match output is 0 it means that PDE worked fine and if match output is 1 then it means that PDE is not detecting the metal patterns correctly. Regressions are setup to automate this testing of PDE regularly.
The ruby script can generate single, dual and tripple patterns, which increases the number of test cases to validate PDE engine, making the test more vigorous and efficient.

Below is the image of single and dual patterns in the gds format generated by the script.

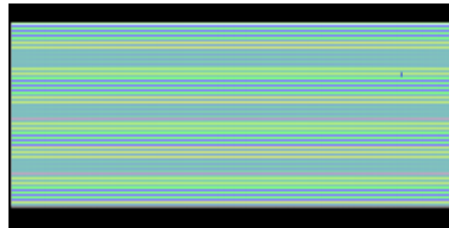

Figure 3.6: Single arrangement metal pattern generated by the script

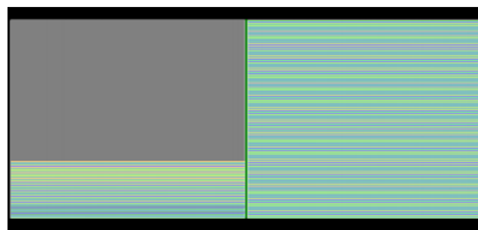

Figure 3.7: Dual arrangement metal pattern generated by the script

Chapter 4

Results

4.1 Fill using Bounding Box

The development of bounding box enables the user to do metal fill in certain regions as per the density requirements in the local neighbourhood of the layout. The regions to be filled can be specified by the user through coordinates of the polygon they want to fill. The command to do fill in a rectangle can be given as below:

--bbox "{{x1,y1},{x2,y2}}"

Here, x1,y1 denote the lower left coordinate point of the rectangle and x2,y2 denote upper right coordinate of the rectangle. The output of doing Metal-2 fill in a bounding box region of {{0,0},{5,5}} looks like below:

Figure 4.1: Rectangular metal fill done using bbox command

User can also use this capability to do a fill in a polygon of any shape. For example to do fill in a triangular region, user can provide bounding box coordinates as below:

-bbox "{{x1,y1},{x2,y2},{x3,y3}}"

The output of doing fill in bounding box with coordinates {{2,2},{5,2},{3,4}} looks like below:

Figure 4.2: Triangular metal fill done using bbox command

4.2 Incremental Fill to reduce manual efforts

As already discussed above incremental fill is extremely useful when used with bounding box capability to reduce manual efforts.

Without implementation of incremental fill first instance of metal fill would be lost if metal fill is run again on the design already having metal fill. Below figure shows the output of running metal fill in region {{6,6},{10,10}} for the second time on the output of running metal fill in a bounding box of {{0,0},{5,5}} shown in the without using incremental fill capability.

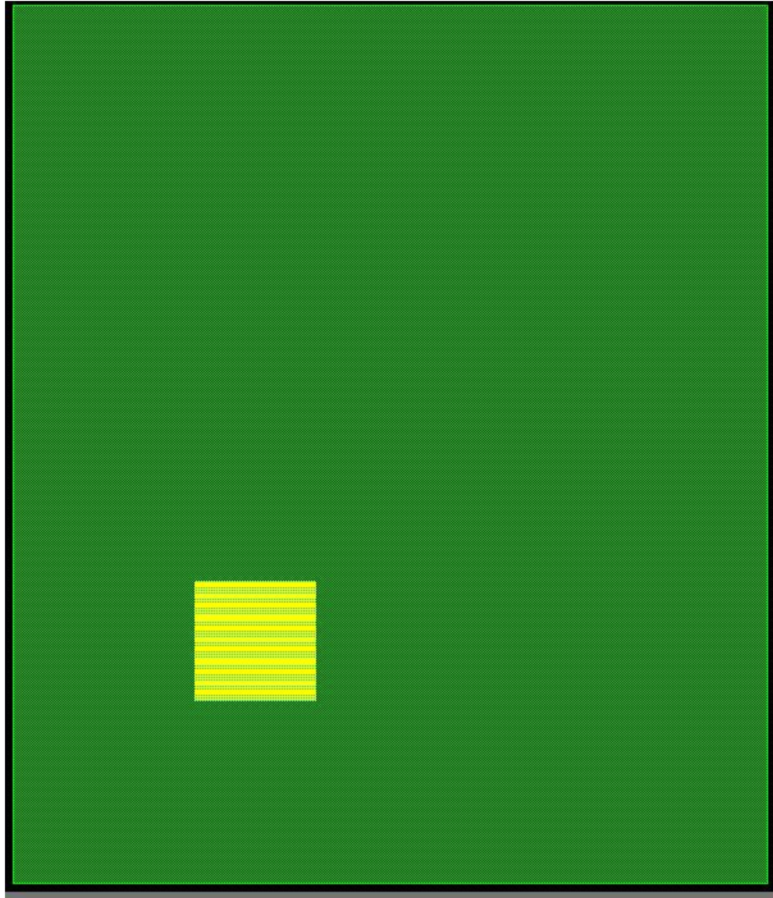

Figure 4.3: Second instance of metal fill done in absence of incremental fill

It can be seen in the above figure that the metal fill in the {{0,0},{5,5}} region which was done previously is replaced by metal fill in region {{6,6},{10,10}} which is not what was expected. Metal fill should have been present in both the regions.

Below figure shows the output of running metal fill or second time in region {{6,6},{10,10}} with incremental fill enabled in the flow. Here, you can see that both the regions are filled with metal layers in the output.

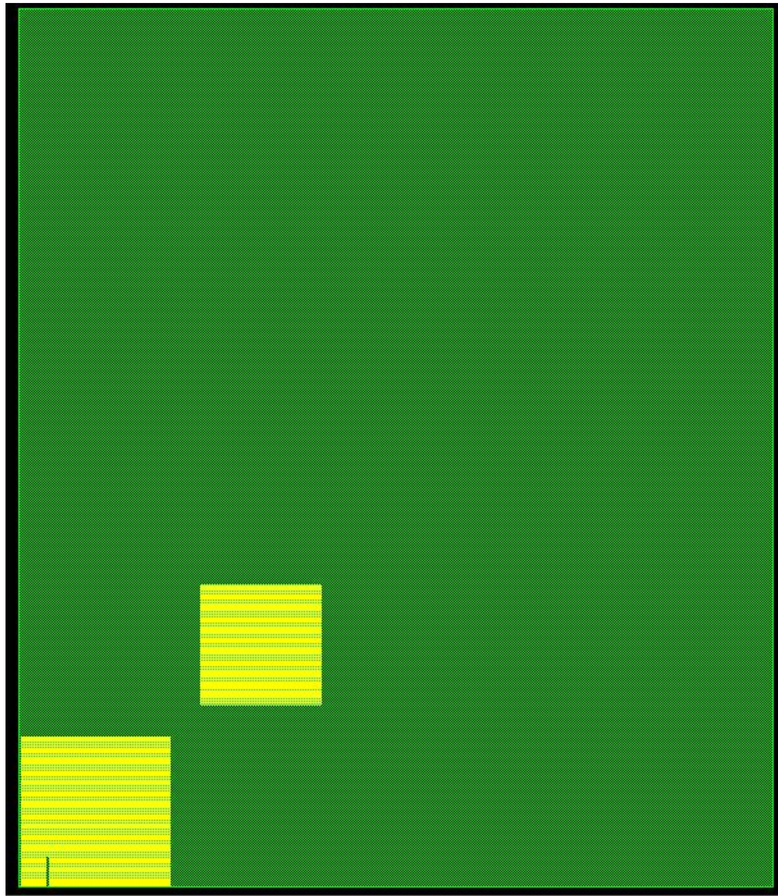

Figure 4.4: Second instance of metal fill done with incremental fill enabled

4.3 Result of implementing new PDE Flow

PDE was able to resolve the minimum density issues in the design. Apart from this it was also able resolve some DRC's which came up earlier due to inconsistent and random transition patterns being placed between existing metal signal lines. Below are a few examples of DRC's solved.

Mx ** : Only Rectangular Metal x shape is allowed, and only allowed in OGD (Orthogonal to gate direction). Earlier metal-ll was not able to detect the underlying Mx patterns and so placed another Mx metal tracks over it as transition patterns causing shorts. This issue was solved with new PDE ow

End-to-end space between metal tracks completely aligned should be more then xx. This condition was added in as a constraint to CPLEX optimizer while determining the transition patterns. As a result the all metal tracks aligned end-to-end would always have a minimum xed distance between them.

However, after integrating the PDE flow with fill API we saw some glitches in the overall metal fill run-time. PDE was taking more time to detect underlying patterns in some complex testcases. Below is the image of runtime degradation observed in metal fill stage after implementation of new PDE flow.

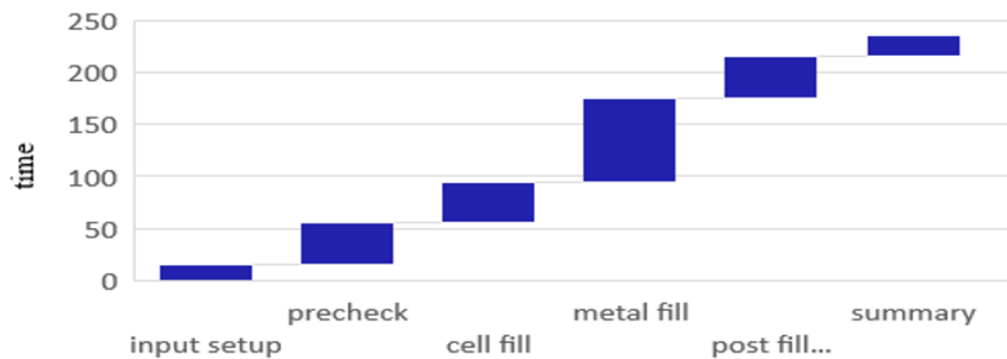

Figure 4.5: Runtime degradation in metal fill stage

Chapter 5

Conclusion and Future Scope

To get higher yield of MOS IC chips, CMP plays an important role. To ensure CMP does its job efficiently, base fill and metal fill layers have to added to the input layout design.Adding dummy filler cells into the design ensures there is no empty space in the design and thus makes sure that there is no density violation occuring. Apart from this fill also plays an important role in Design rule check.Fill is capable of removing the DRC's which were present is the design pre-fill. For proper metal-fill in the design PDE is required so that there are no other violations with respect to already existing interconnects in the design. This requires proper testing of PDE and so more and more test cases are required to make testing efficient. After the development of PDE, designers could easily do metal fill in the whitespace regions in their design to meet density requirements without having to worry about LVS or DRC violations. This helped them in closing their design on time during the sign off stage of the project.

However, as of now since metal-fill is taking more time as compared to other fill stages, in future new techniques would be developed that can help reduce the over-all fill run time.

References

[1] L. Nan, K. Mouthaan, Y. Xiong, J. Shi, S. C. Rustagi, and B. Ooi, “Improved microwave modeling of cmos spiral inductors with metal dummy fills,” in *2008 10th Electronics Packaging Technology Conference*, pp. 275–278, Dec 2008.

[2] V. S. Shilimkar and A. Weisshaar, “Modeling of metal-fill parasitic capacitance and application to on-chip slow-wave structures,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 65, pp. 1456–1464, May 2017.

[3] A. B. Kahng and K. Samadi, “Cmp fill synthesis: A survey of recent studies,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, pp. 3–19, Jan 2008.

[4] B. E. Stine, D. O. Ouma, R. R. Divecha, D. S. Boning, J. E. Chung, D. L. Hetherington, C. R. Harwoo, O. S. Nakagawa, and Soo-Young Oh, “Rapid characterization and modeling of pattern-dependent variation in chemical-mechanical polishing,” *IEEE Transactions on Semiconductor Manufacturing*, vol. 11, pp. 129–140, Feb 1998.