EAI Endorsed Transactions
on Cloud Systems

Research Article EAI.EU

Layout Automation Techniques to Optimize Time-to-Market Factor

G S Aishwarya Meghana[1], Sheetal Y Kochrekar[1] and Poornima Venkatasubramanian[1,*]

[1]Design Enablement (DE), Samsung Semiconductor India Research (SSIR), Bangalore, India

Abstract

With the ever growing demand for IPs in the domain of mobiles, 5G, and sensors for wearables and IoT applications, time to market becomes a crucial factor. There is always a need for delivering IPs to the competitive market in very less time without compromising on the quality. This necessitates the development of a reliable automation technique that can be easily integrated into the design cycle and Quality assurance flow of an IP. In this paper, we propose a generic automation flow for generating seed layouts for any technology node. The proposed flow reduces manual efforts and improves overall productivity.

Keywords: Automation, Layout, Time to Market, IP, Sensors, IoT, Mobiles.

Received on 02 June 2019, accepted on 29 June 2019, published on 16 July 2019

doi: 10.4108/eai.16-7-2019.162216

[*]Corresponding author. Email: p.venkatasub@samsung.com

1. Introduction

As the demand for Application Specific Integrated Circuits (ASIC) at lower technology nodes like 14nm, 10nm, and 7nm increases, foundry DRC rules become more complex and demands more time to generate layouts manually. This has a huge impact on the time-to-market factor for any industry.
Both fab and fabless industries compete in the fast-growing domains like AI, IoT and mobiles to acquire major share in the market. Automation techniques can minimize time-to-market tremendously by assisting the VLSI designers to adhere to schedules for timely delivery to the market.

Standard cells and Memories occupy 20% to 60% of any System on Chip (SoC) depending on the application. Area optimization of Standard cells and Memories can have a huge impact on the total area of the chip. There are multiple area optimization techniques implemented both in Circuit design and Layout design. Area estimation of any block at an early stage of design cycle is of key importance to develop an efficient layout with quick turn-around time.

We have developed a generic layout automation flow that can assist layout engineers to automate the layouts. The layouts generated through this flow are LVS clean and base DRC clean. Routing can be custom to optimize the capacitance and resistance for timing specifications.

In this paper we focus on Standard cells and Memory compilers to demonstrate the advantages of the proposed Automation technique. Section [II] explains the fundamentals of placement strategies which were explored. Section [III] explains the methodology of the proposed automation technique. Section [IV] demonstrates the Experiments and Results. Section [V] explains the scenarios in which flow can have tremendous impact along with the Future scope of the proposed automation flow.

2. Literature survey

There are multiple CAD based techniques to automate the entire Layout flow after the design phase in ASIC/IP design cycle. Cadence Layout XL provides the option for the automated placement and routing options without Pcell customization. Various placement algorithms with the intention of optimizing the area and delay of the critical signals through minimum routing techniques were presented in earlier works [2]. Standard cells have their height fixed and width varying based on functionality. The standard cell structure allows to automate based on the fact that it has a very well defined cell height, horizontal power bus and fixed power bus widths. Thus any module with large number of standard cells are very convenient to automate. Row-wise placement is followed in this case. Module height will be in multiples of height of the standard cell. This layout is inherently optimized for minimum area and routing criterion [3]. We have shown the generic Physical design flow for any IP in the flowchart below.

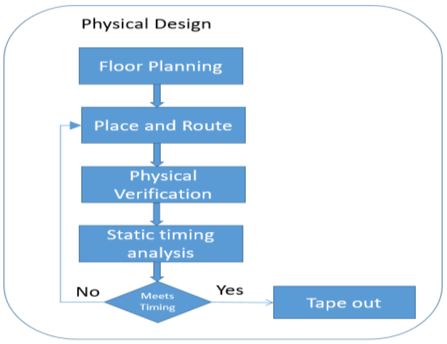


Figure 1. Generic Physical design Flow of an ASIC/IP

3. Methodology

The generic flow that we have developed can be used for generating layouts of any technology nodes. To demonstrate the impact of this flow, we discuss only Standard cells and Memory compilers in this paper.

3.1 Overview of the flow

A GUI is developed in SKILL language to provide user interface for the proposed flow. SKILL commands help to analyze the Cadence virtuoso database and direct the flow based on the user inputs.

The proposed flow has the option to choose Standard cells and Memory compilers. The placement algorithms for both choices are unique and customized. The control block of Memory Compilers schematic has a hierarchical structure unlike the standard cell circuit and hence a different placement algorithm is employed. The layout automation flow reads the input netlist or schematic and generates its equivalent layout that is LVS clean and base DRC clean. The flow does not provide automated routing options.

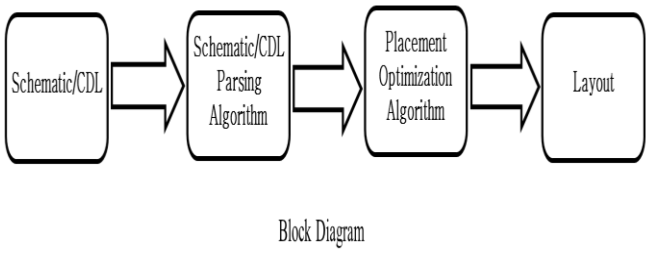

Schematic/CDL
Schematic/CDL Parsing Algorithm
Placement Optimization Algorithm
Layout
Block Diagram

Figure 2. Overview of the proposed Automation flow

3.2 Inputs to the flow

3.2.1 Parameterized cell (Pcell) customization file

Pcells are provided as a part of PDK from the foundry to the design team. Pcells of basic MOSFETs are customized based on the position of the MOSFET in the layout. Basic customization of Pcells which is sufficient for any standard cell circuit is given below. CT layer defines that Gate(poly layer) should not be shared. For every standard cell, the boundary device has this CT layer at the cell outline.

t

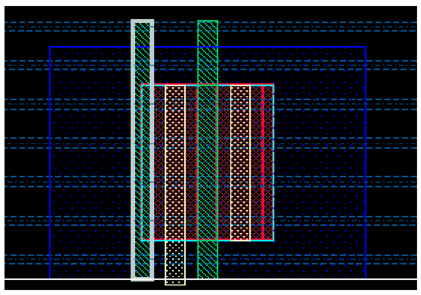


Figure 3. Pcell of FET with CT on the left

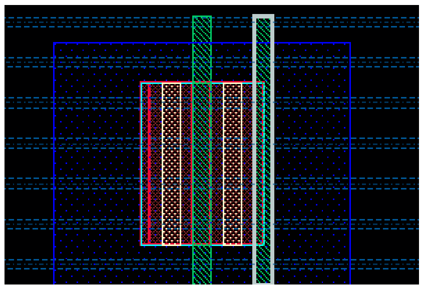


Figure 4. Pcell of FET with CT on the right

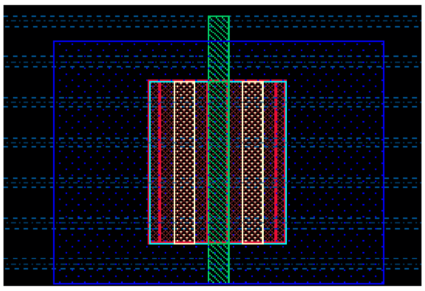


Figure 5. Pcell of FET with no CT.

Based on the placement strategy, a suitable Pcell is assigned to the device in the layout. Users are given the flexibility to customize Pcells based on their requirement and input to the flow. The below image shows how three different Pcells with different CT combinations are placed based on the connectivity.

3.2.2 Technology node, Poly pitch (CPP), Power bus width, device offset, maximum number of device fins, Metal tracks.

These parameters are used for calculating the base DRC rules while placing the Pcells in the layout and overlapping the Pcells within DRC limits when two devices drain/source is shared. Metal tracks value is utilized in calculating standard cell height. Maximum number of device fins input will be used to split the larger device into fingers (parallel devices) according to the cell height of the standard cell.

3.2.3 Top cell width and leaf cells width in Control block

These inputs are used for Memory compilers. This provides the users flexibility to choose the width of each block in the hierarchy of the intended block along with its total width.

3.2.4 Netlist

Spice netlist of the schematic is input to the flow when you choose the option to generate Standard cell library in Netlist mode.

4. Flow for Standard Cells

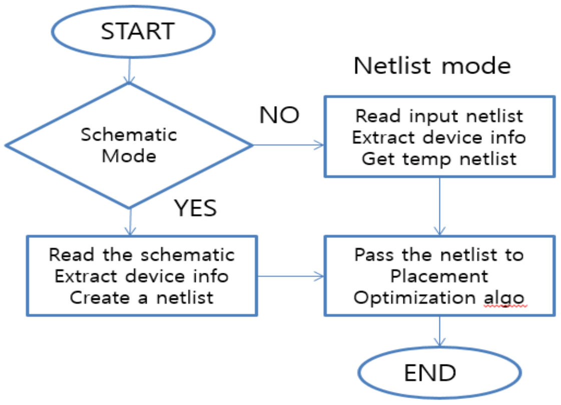
For standard cells, the flow provides two modes.

1) Schematic mode
2) Netlist mode

In both the modes, the tool reads the input, generates temporary netlist and utilizes a placement optimization algorithm to give the most optimal placement. This acts as input to Layout generation. The layout will have an optimal placement.

A) When the schematic view is input to the flow, the first step is to read the schematic and identify the details of the devices, pins and interconnections present in the schematic. From this information, a netlist file is generated.

B) When the Netlist is input to the flow, device information of standard cells along with the connectivity information is parsed and analyzed to find an optimum device placement strategy to get an area efficient layout.



4.1 Placement Optimization Algorithm

Utilizing the concept of stick diagram to optimize the area of standard cell, the automated layout will be same as that of manually generated layout [1]. Any commonly used standard cell like NAND, NOR, INV or Flip-flop will have the output connected to CMOS pair transistors. Hence determining this transistor pair is the starting point of the proposed algorithm.

Step I: Determine the last pair of transistors to be placed in the layout i.e. the complementary transistor pair with common gate whose drains are connected to output pin.
In case of cells with more than one output pin, the tool considers the output pin that it encounters first in the CDL file/schematic.

Step II: All other transistors are arranged in some order by checking a list of conditions so as to minimize the area of the layout. The conditions that are checked for placing a transistor pair are:

1) Number of fins per finger are same as that of the previously placed transistor (both NFET and PFET) (High priority)
2) Possibility of sharing drain/source with the previously placed transistor (both NFET and PFET)
3) The NFET and PFET have the same gate
4) Gate of NFET and PFET are same as that of the previously placed NFET and PFET respectively. (Low priority)

There is a priority involved in checking the conditions specified above whenever it is not possible to find a pair of transistors satisfying all the four conditions. There is a separate algorithm designed for sequential cells, Flip-flops, where there is one more condition involving the possibility of cross coupling the pass transistors with complementary gates. This algorithm takes into account the master-slave topology for device placement.

Once the order of the transistors is determined, the order of drain and source for each transistor is determined. Finally the transistors are placed in the layout using Pcells.

4.2. Example of the Algorithm

Consider the case of automatic generation of layout for the AND gate. The logic and the anticipated layouts after each of the iterations performed by the tool for generation of layout are shown in below

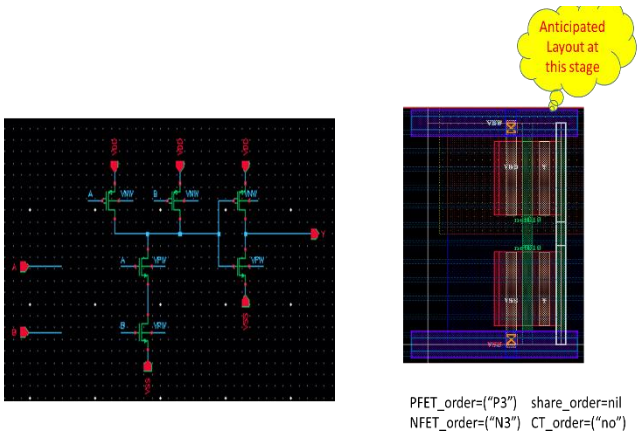

Anticipated Layout at this stage
PFET_order=("P3") share_order=nil
NFET_order=("N3") CT_order=("no")

Figure 5. Anticipated Layout after the 1st stage

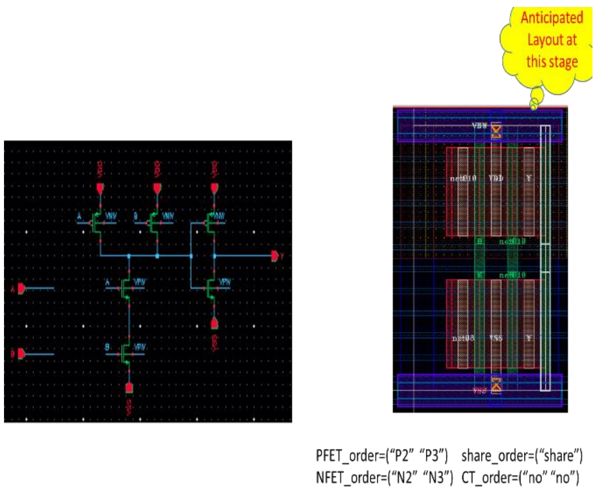

Anticipated Layout at this stage
PFET_order=("P2" "P3") share_order=("share")
NFET_order=("N2" "N3") CT_order=("no" "no")

Figure 6. Anticipated Layout after the 2nd stage

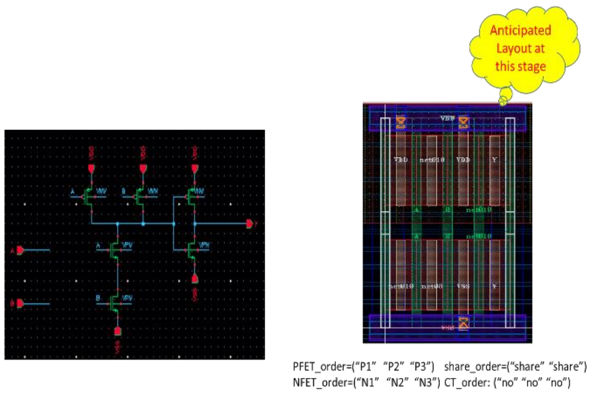

Anticipated Layout at this stage
PFET_order=("P1" "P2" "P3") share_order=("share" "share")
NFET_order=("N1" "N2" "N3") CT_order: ("no" "no" "no")

Figure 7. Anticipated Layout after the 3rd stage

5. Flow for Memory Compilers

For Memory compilers with more complex schematic hierarchy, the proposed flow employs a unique approach to generate layout of the same hierarchy. The flow is enabled to take schematic view as input and parse the schematic to understand the hierarchy.

Top-down and Bottom-up approaches are used to generate layouts of control block in Memory compilers. When schematic view is input, the flow descends the hierarchy until the standard cells are found and generates the layouts of all standard cells in the entire hierarchy. The flow enables the random placement approach and generates the layouts of every instance in the hierarchy. Further, bottom-up approach is employed to utilize the generated layouts of all the hierarchy levels to generate the layout of top-level input schematic. The flow provides the user with the flexibility to change the placement of the standard cells in any hierarchy level.

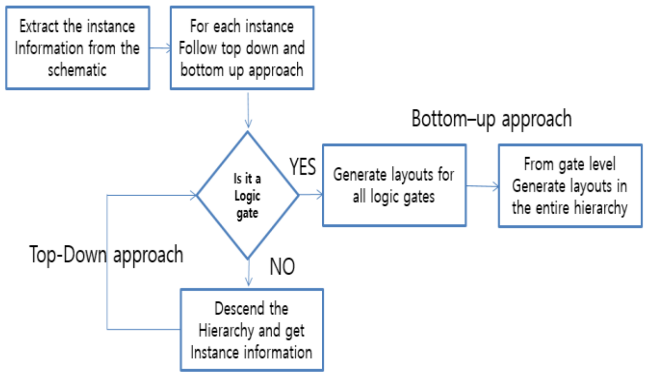

Extract the instance Information from the schematic
For each instance Follow top down and bottom up approach
Is it a Logic gate
YES
NO
Bottom–up approach
Generate layouts for all logic gates
From gate level Generate layouts in the entire hierarchy
Top-Down approach
Descend the Hierarchy and get Instance information

Figure 8. Overview of automation flow for schematic with hierarchy

The layouts generated with this flow has all the pin level information and hence can preserve the connectivity information when the layouts are opened in Cadence Layout XL. Based on the connectivity information the users have the flexibility to change the placement of the blocks according to the requirement.

6. Experiments and Results

The proposed flow is implemented for both standard cell libraries and Memory compilers.

6.1 Ultra high speed Register file memory compiler

A Memory instance contains modules like Bitcell array, IO, decoder and Control block. Layout for IO and decoder is custom and smaller in terms of area. Automated floor planning techniques for memory compilers help in replicating these IO and decoder multiple times in a memory instance based on the configuration. SRAM 6T/8T Bitcell layout is often provided by the foundry PDK. Bitcell array is tiled using automated floorplan. Control block of the memory instance contains modules like Clock generator, Tracking, and other timing critical circuits for High speed memories. Since control block is critical in terms of area and timing, it often demands more time to generate layout. We have verified the proposed flow to automate the seed layout of control block of ultra-high speed memory. Time taken for handmade seed layout when all standard cell layouts are available is 2 weeks whereas, the flow can generate the seed layout of entire control block with multiple hierarchies including the standard cells in less than 10 minutes.

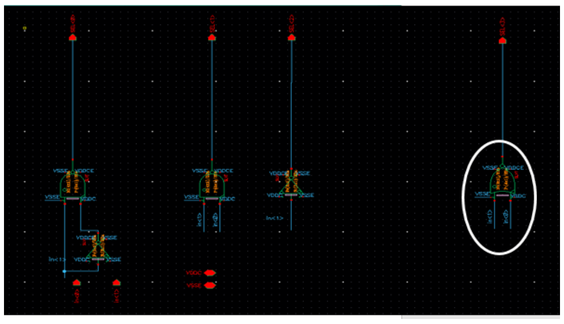


Figure 9. Schematic with multiple standard cells.

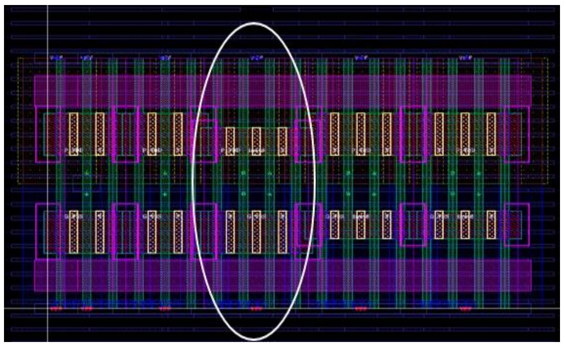


Figure 10. Optimum Layout of the schematic shown above.

The Fig. 4 shows the schematic of the sample module with the multiple standard cells at some hierarchy level of the control block of the chosen memory instance. The Fig. 5 shows the layout generated through this proposed flow.

6.2 Standard cell libraries used in SERDES and Image Sensor IPs

Standard cell libraries usually contain logic gates like NAND, NOR, INV, AND, OR and few sequential circuits like D-Flipflops with multiple variations based on drive strengths. IPs like SERDES, Image Sensors have most of their chip area occupied by standard cells. Standard cell libraries are optimized for leakage, power, performance and area depending on the specifications provided by the customer. Standard cell delivery kit usually contains netlists and layouts of cells along with their characterization data. Standard cell libraries design cycle involves multiple iterations of circuit design and Layout design to meet the required specifications. Layout automation assist to quickly develop libraries and characterize for Power, Performance and Area.

To demonstrate the impact of the flow we have chosen a Standard cell library with 800 cells. This library has multiple variants of all the commonly used logic gates. Time taken to develop this library is very much less through this proposed flow.

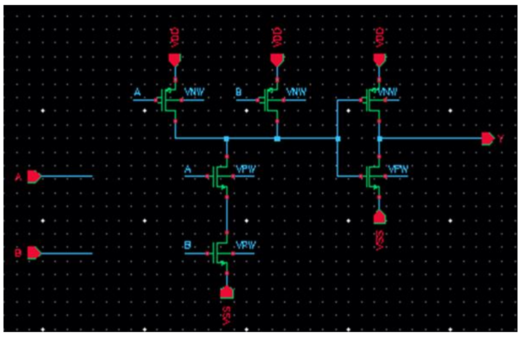


Figure 11. NAND-INV circuit schematic

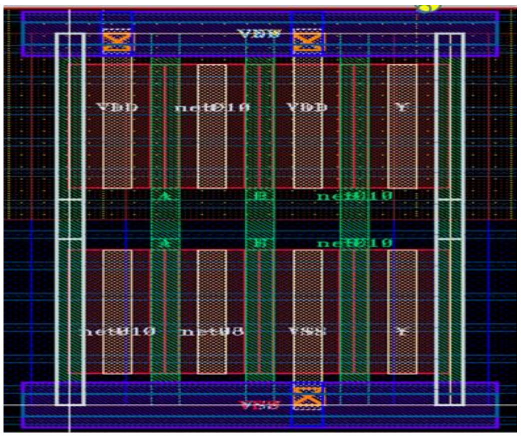


Figure 12. NAND-INV circuit layout

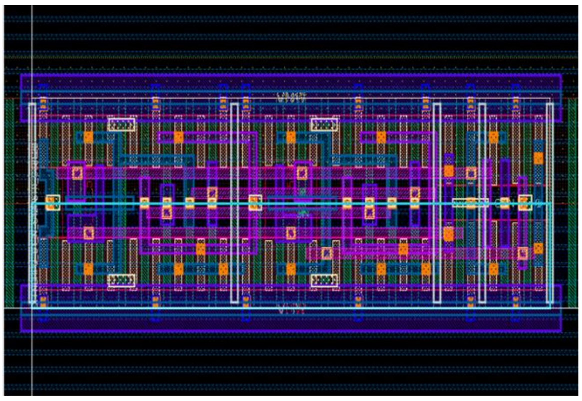


Figure 13. Handmade layout of a Flipflop with routing

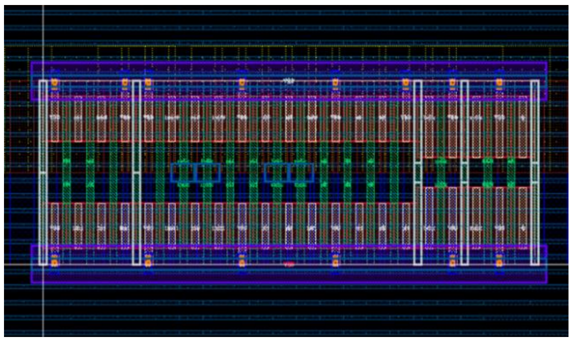
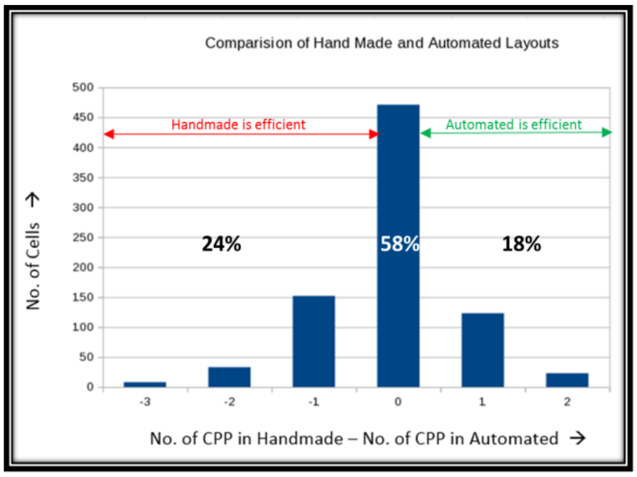


Figure 14. Automated layout of a Flipflop without routing

The Fig. 6 shows the schematic of AND gate in the chosen standard cell library. The Fig. 7 shows the layout of the corresponding AND gate generated through this proposed flow. The Fig. 8 shows the layout of D Flipflop generated manually. The Fig. 9 shows the layout of the same D Flipflop generated through the proposed flow.



- Cells considered for comparison - ~800
- Layout execution time with tool - ~10min

Figure 15. Comparison of Handmade layouts & Automated layouts

The graph in Fig. 15 indicates that 58% of the cells have same area as that of the manual layouts and 18% of the cells have lesser area when compared to that of manual layouts. Only 24% of cells have more area than that of manual layouts and since these are seed layouts, manual intervention can optimize the area. The above stated data does not consider the routing complexity factor.

7. Advantages and Future Scope

7.1 Advantages of the flow

- User flexibility to develop the layout at any hierarchy with the desired width in standard cell pitch.
- The flow can be utilized for both CMOS and FINFET technology.
- User flexibility to customize the Pcells.
- Layout XL compatibility for custom placement of complex blocks.
- Improves the overall productivity by minimizing the cost in terms of resources and increasing the efficiency.
- This tool does not require any extra purchase of licenses

7.2 Scenarios

In this section, we will discuss few scenarios where the proposed tool becomes handy.

7.2.1 Quick area estimation of critical blocks with multiple modules

Area estimation of the critical blocks at the early stage of development cycle would provide an insight regarding the module that has to be focused to optimize the area. The proposed flow allows the user to adjust the aspect ratio of any module. Thus the proposed flow helps to optimize the layout in multiple iterations in very minimal time.

7.2.2 Standard cell Architecture analysis

Standard cell architecture definition includes identifying the maximum number of allowable metal tracks, power bus widths and positions. Automation can help in reducing the time taken for architecture evaluation by providing the user with the flexibility to change inputs and quickly characterize the cells. Once the architecture is finalized automation can tremendously reduce the time taken to develop any standard cell library.

7.2.3 First cut layouts of any schematic with complex and multiple hierarchies

The flow can descend the entire hierarchy and generates the layouts at each level. This helps the layout engineer to understand the modules to be focused to optimize the routing complexity and robustness. Eventually the user can spend more time on sign-off verifications like EM/IR estimation. Thus the flow minimizes the time taken for the iterative process of optimization of complex blocks for Power, Performance and Area (PPA).

7.2.4 PDK evaluation

For latest technology nodes the PDK from the foundry has to be evaluated for specific logic structures like NAND-NOR, NAND-INV and NOR-INV. This automation flow can help to quickly generate layouts with different standard cell architectures to analyze for required PPA. Then the best optimized cell structure is chosen.

7.3 Future scope

- Current Placement strategy of modules in schematic with complex hierarchy is random. The tool can be customized to take the user inputs to direct the placement of certain critical blocks.
- Extension of the flow to support Analog and Mixed signal layouts. This includes developing the placement strategy for Analog layouts that would requires perfect matching. Common centroid layouts are immune to cross chip gradients and provides best possible matching. Common centroid layout technique can be included in the flow.
- Concept of machine learning can be included in the flow to help in custom blocks which does not follow standard cell grid. In this procedure, the flow would be able to learn from the input handmade optimized layouts.

8. Conclusion

In this paper we have demonstrated the impact of the flow on the development time frame of the Standard cell libraries and Memory compilers. As discussed in the paper, the proposed flow minimizes the number of resources required and time-to-market. Thus minimizing the overall costs and increases the time frame available for layout optimization and validation.

In conclusion, the proposed flow can minimize the manual effort to generate seed layouts, perform analysis and Area estimation. The proposed flow can be customized according the user needs. In this paper, we have discussed the future scope of the flow to make it more generic and suitable for layout automation in various domains. Shrinking the required time to generate seed layouts at the early stages of design cycle will improve the productivity by developing more robust products in less time frame. This is most suitable in the current scenario of market which has very high demand for IPs in the domains like Mobile, 5G, Sensors and AI.

Acknowledgement

We would like to thank Parvinder Kumar Rana for his continuous support and help in this journey. We would like to thank Ravi

Kumar Gajula for his constant guidance in developing the automation technique.

References

[1] RABAEY, J. M.; CHANDRAKASAN, A. P.; NIKOLIC, B. Digital integrated circuits. [S.l.]: Prentice hall Englewood Cliffs, 2002.

[2] K. SHAHOOKAR AND P. MAZUMDER, "VLSI cell placement techniques," Journal, ACM Computing Surveys (CSUR), Volume 23 Issue 2, pp.143-220, June 1991.

[3] Tokinori Kozawa, Hidekazu Terai, "Reseach in Design Automation for VLSI Layout, " *IEEE Design & Test of Computers*, pp. 43-53, October 1985.

[4] "Cadence Virtuoso Skill Language Reference," Product Version 1CADV12.2, June 2016.

[5] "Cadence Virtuoso Parameterized Cell Reference," Product Version ICADV12.2, July 2016.