A Simplified Methodology for Complex Analog Module Layout Generation

Pradeep Kumar Chawda
Texas Instruments Inc., 3833 Kifer Rd, Santa Clara, CA
E-mail: pkc@ti.com

Abstract

Analog layout design automation has been evolving constantly and several attempts have been made to find a solution for analog synthesis. Due to the complexity of analog design problem it is difficult to find a single approach which can be readily accepted by the industry. The optimization based full analog synthesis tools are quick but does not capture layout engineers' expertise and therefore produce sub optimal layout. The semi-automated layout tools help layout engineers considerably however the layout creation cycle time is still at unacceptable level for time-consuming analog designs.

This paper presents a simplified methodology for generating complex analog modules layout using template driven parameterized cells to reduce layout creation cycle time significantly while meeting the layout designers need. The paper discusses step by step approach of developing a placement and routing template to capture layout engineers' expertise for complex analog modules and demonstrates its effectiveness by implementing a triple cascode amplifier super-pCell which is being used in pipeline Analog to Digital Converts. The implemented methodology is very flexible and fully controllable so that designers can easily create a layout with additional design requirements and constraints quickly. The proposed approach is successfully adopted by layout engineers and as a result, the required layout resources for a design are reduced significantly whereas layout engineers' efficiency is improved significantly.

Keywords

Analog layout automation, Super pCell, Template driven layout, Analog Synthesis

1. Introduction

Modern day integrated circuits provide system on chip (SoC) functionality, which requires the co-development of both analog and digital blocks and the integration of these blocks to create a system. The plethora of CAD tools available to designers of digital blocks for these SoC's allows the designer to push a digital design in to the layout phase at a much faster rate than analog blocks. The push for higher levels of integration on chip places pressure on analog designers to reduce the development time of their analog circuits. Possibly the most tedious and time-consuming aspect of analog circuit design is the layout phase. Traditionally, each polygon had to be drawn by hand in order to realize layout. Now, complete devices or even complete blocks can be drawn automatically [1].

Analog layout creation can be divided in following five phases.

1. Circuit partitioning and constraint generation, consisting of a) identifying analog blocks (e.g. core, bias, critical non critical, noise sensitive/high current/power blocks etc.) and b) generating constraints for device generation, placement and routing
2. Device identification and device generation, and includes a) generating discrete components MOSFETs, resistors, inductors and capacitors taking into account aspect ratio, symmetry, matching and process variations, b) generating stack MOSFETs, composite MOSFETS (current mirrors etc.) and c) generating guard rings/substrate contacts etc.
3. Placement subject to the placement constraints identified in step 1 such as X mirrors/Y mirrors/near to edge/minimum parasitic/thermal constraints etc.
4. Routing subject to the routing constraints identified in step 1 such as symmetry/matching/minimum parasitic/electro migration/crosstalk aware etc.
5. Verification and optimization consisting of Design Rule Check (DRC), Layout Vs Schematic (LVS) and optimization for area and performance.

There are multiple iterations involved before the layout is finalized. First circuit designers create design schematic and wait for layout engineers to provide design layout. Next circuit designers extract parasitics from layout using extraction tools and simulate the design to validate if the layout still meets the design specification. In most cases circuit designers update the device sizes to meet all the design specification which triggers another round of design layout. After several iterations, the layout is finalized. The frequent changes in device sizes affect the placement and routing adversely. Meeting the area and performance constraints simultaneously becomes very difficult. For advanced technologies, the technology constraints (Design for Manufacturability) make layout creation even more challenging.

Several approaches have been proposed [1] over the last decade to automate the analog layouts. One can refer to generator based [1-4], template based [5-6] and optimization based [7-10] approaches. Optimization can handle the constraints explicitly [10-13] while generators implement export knowledge implicitly. There are hybrid approaches as well [14-18]. One of the challenges for adoption of these approaches is steep learning curve required by layout engineers to effectively use these analog layout automation tools.

This work proposes a simple, fast, intuitive, easy to use, highly controllable and flexible methodology for analog

layout creation which significantly reduces overall analog design cycle time. The approach is inspired by template-based methods which provide high level layout as well as detailed routing constraints. This paper presents a template driven parametrized cell methodology for generating complex analog module layout. The presented methodology is engineering change order friendly and flexible so that designers can create layout with additional design requirements and constraints on the fly. Section 2 explains the methodology to develop placement and routing template for writing mega pCell. Results are presented in section 3. Section 4 draws the conclusion.

2. Template Driven Parameterized Cell Methodology

To demonstrate the methodology, an analog module from an Analog to Digital Converter (ADC), a triple cascode amplifier is identified. An amplifier is a basic analog block, which is used several times in an ADC design and can be shared among several ADCs with slight modification. Automating the layout creation of amplifier block substantially reduces the amount of time and effort spent in creating these layouts for all the ADCs. The amplifier layout is generated as super parameterized cell and any changes if required can happen in fraction of a minute. The parameterized cell (pCell) is implemented in Cadence's SKILL language [19-20] however other languages such as Python and frameworks can also be used.

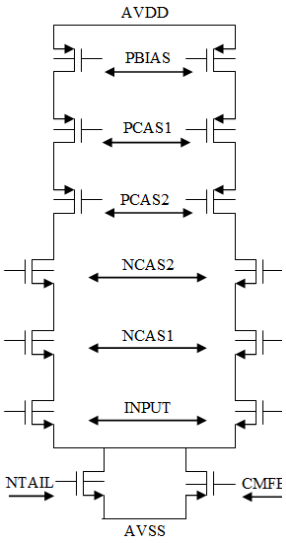


Figure 1: Triple cascode amplifier excluding biasing and common mode feedback schematic

The complete framework for using super pcell consists of following components:

- Graphical user interface for controlling pCell parameters
- Identifying the schematic based on given template.
- Importing data from schematic for generating pCell.
- Calculating default values for pCell parameters.
- Generating pCell for three different layout aspect ratios.
- Area and performance optimization.

To summarize, first analog modules which can be parameterized are identified. Next a placement and template is created considering all constraints. Finally layout pCell is coded either manually or using GUI pcell creator. The triple cascade super pCell development is discussed in detail in following subsections. The approach of identifying placement template and routing algorithms are applicable to other analog modules as well and will be useful in creating similar or even bigger pCells for automating analog layout. Advanced layout constraints such as electromigration, IR drop, DFM can be taken care of in placement and routing template and therefore in the generated super pCell itself.

2.1 Template creation for placement and routing based on design constraint

From the schematic it can be seen that the Super pCell can be developed as two half mega pcells as it has two identical paths from AVDD to AVSS. The half-cell is used to create full amplifier layout by placing the half-cell twice with R0 and MX orientations. The schematic selected for creating amplifier pcell is shown in the Figure 1. The AVDD and AVSS are net names. NTAIL, CMFB, INPUT, NCAS1, NCAS2, PCAS1, PCAS2 and PBIAS are transistor names. All NMOS substrates are connected to AVSS and all PMOS substrates are connected to AVDD. Each transistor is placed by splitting it in multiple rows and columns. For creating the layout, length, width, finger and

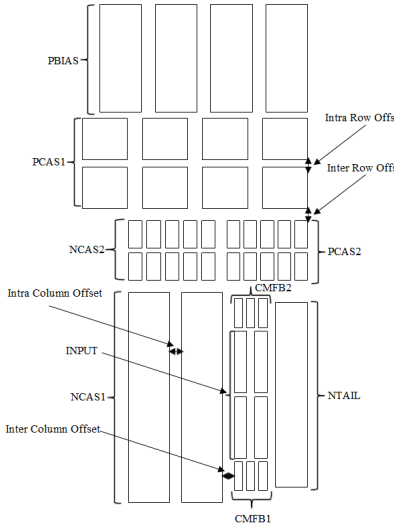


Figure 2: Layout floor plan illustrating transistor placement for half-cell for schematic of figure 1.

Table 1: Brief description of transistor placement parameters

Name	Cell Name	Instance Name **(Fixed)**	Gate **(Fixed)**	Gate Placement **(Fixed)**	Rot. F
NTAIL	NCH4T_1P8V	NTAIL_MAIN(R)_(C)	1 per device	Right Side	R0
CMFB1	NCH4T_1P8V	CMFB1_MAIN(R)_(C)	2 per device	Left &Right	R0
CMFB2	NCH4T_1P8V	CMFB2_MAIN(R)_(C)	2 per device	Left &Right	R0
INPUT	NCH4T_1P8V	INPUT_MAIN(R)_(C)	2 per device.	Either Side User control	R0
NCAS1	NCH4T_AF_1P8V	NCAS1_MAIN(R)_(C)	2 per column	Opp. sides	R0
NCAS2	NCH4T_1P8V	NCAS2_MAIN(R)_(C)	2 per device	Left &Right	R0
PCAS1	PCH4T_1P8V	PCAS1_MAIN(R)_(C)	2 per column	Left &Right	R0
PCAS2	PCH4T_1P8V	PCAS2_MAIN(R)_(C)	2 per column	Left &Right	R0
PBIAS	PCH4T_3P3V	PBIAS_MAIN(R)_(C)	2 per column	Left &Right	R0
Note: A cell is a row & column split of a transistor. R is used for row & C for column.					

Name	Intra Col. Offset X(um) **(Fixed)**	Intra Row Offset Y(um) (Fixed)	Inter Xtor Offset X(um)	Inter Xtor Offset Y(um)	No of Substrate/cell (Fixed)	Substrate Placements (Fixed)	Type of Substrate (Fixed)
NTAIL	10.0	-1.0	5.0	5.0	1	Right Side	Separate
CMFB1	1.2	2.0	0.0	-5.0	2	Top, Bottom	Butting
CMFB2	1.2	0.0	0.0	5.0	2	Top, Bottom	Butting
INPUT	1.2	-1.0	3.5	5.0	1	Right Side	Separate
NCAS1	10.0	0.0	15.0	-5.0	2	Left & Right	Separate
NCAS2	1.2	4.0	5.0	0.0	2	Top, Bottom	Separate
PCAS1	1.2	2.6	0.0	15.0	2	Top, Bottom	Non-Butt.
PCAS2	1.2	1.8	0.0	20.0	2	Top, Bottom	Butting
PBIAS	7.4	12.2	3.7	10.0	2	Top, Bottom	Butting
Note:	The red color shows parameters, which cannot be changed later by user while green color parameters are customizable by user after creation of pcell.						

transistor cell type is taken from schematic while number of rows and columns for every transistor and metal widths and offsets will be taken from super pcell GUI form. Number of fingers in each row/column split is taken as input from a separate form linked with pcell form. All default values used by the super pcell are user controllable through the super pcell GUI form.

The placement and routing template is deduced from existing layout of the amplifier while meeting all design constraints. Figure 2 shows placement template. The routing template as deduced is discussed in the subsequent sections. The template is flexible and can expand or collapse as needed. The template can generate layout in three different aspect ratios. The layout designer's intent as well expertize is also captured in the template.

2.2 Placement of transistors and substrates based on design rule guidelines

1. The transistors are identified based on fixed schematic name and placed in the pcell with name_MAIN and row no and column no. (e.g. NTAIL_MAIN1_1).
2. Transistors are placed in following sequence, NTAIL, LOWER CMFB, INPUT, UPPER CMFB, and NCAS1. Next. NCAS1 INPUT and NTAIL are center aligned. CMFB and INPUT are placed common center. Then PCAS2 and NCAS2 are placed and center aligned.
3. For placing a transistor the X and Y offsets are added to the actual co-ordinates. (e.g. For NTAIL the actual co-ordinate is global origin(0,0), but NTAIL is placed at (0+X_os, 0+Y_os).
4. For other transistors the offsets are relative to the adjacent transistor(s). Apart from length, width, no fingers, no of rows, no of columns, rotation, rout poly and rout substrate, the interdigitated string and routM2finger width is also specified in some of the transistors (which are taken from template).
5. While placing the several splits of transistors, spacing between transistor rows and columns are split offset X and split offset Y.
6. For substrate placement, the number and relative locations are taken from template and specified in the Table 1. There is a ROD object [19], which creates separate substrates. DRC is run to check the sufficiency of the substrate placements as well as contacts. Figure 3 shows the detailed substrate placement using DRC guidelines.
7. Gates are placed as specified in the template and noted in the Table 1.

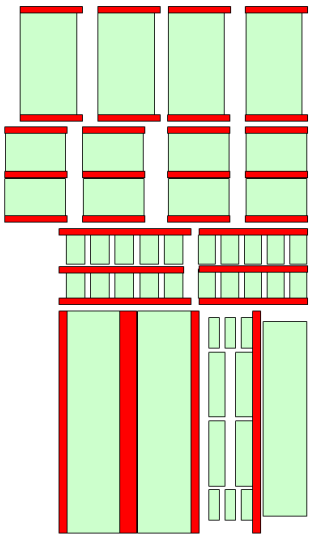


Figure 3: Layout floor plan showing substrate placement

2.3. Routing of intra transistors source and drain

Gate, source and drain connections are decided by the routing template. The routing is done using MET2 for intra source-drain connections; MET3 and MET4 for inter-source drain connection; MET5 for NTAIL source to AVSS connection. For gate and substrate connections MET1 is used. For connecting various transistors source-drain, the metal width spacing and number of metal strips is given in Table 2 and are calculated as follows:

Table 2: Brief description of transistor routing parameters

Name	Met2 Width Min-Max (um)	Met3 Width Min-Max (um)	Met4 Width Min-Max (um)	Met5 Width Min-Max (um)	Gate Met1 Width MinMax (um)	Substrate Met1 Width Min-Max (um)	No of S/D Strips	No of M3 Strips Min
NTAIL	3-10	3-10	3-10	3-16	1.0-2.0	1.0-2.0	2	2S1D
CMFB1	3-10	3-10	3-10	3-10	1.0-2.0	1.0-2.0	2(EC)	2S1D
CMFB2	3-10	3-10	3-10	3-10	1.0-2.0	1.0-2.0	2(EC)	2S1D
INPUT	3-10	3-10	3-10	3-10	1.0-2.0	1.0-2.0	2	2S1D
NCAS1	3-10	3-10	3-12	3-10	1.0-2.0	1.0-2.0	2	2S1D
NCAS2	3-10	3-10	3-12	3-10	1.0-2.0	1.0-2.0	2(EC)	2S1D
PCAS1	3-10	3-12	3-16	3-10	1.0-2.0	1.0-2.0	3	2S1D
PCAS2	3-10	3-10	3-12	3-10	1.0-2.0	1.0-2.0	2(EC)	2S1D
PBIAS	3-10	3-16	3-24	3-10	1.0-2.0	1.0-2.0	3	2S1D
Remarks	Algo.	Algo. M3= 2*M2	Algo.	Fixed	Fixed	Fixed	EC-Routing Style 4b	Algo.

Name	M4 strip	M5 strip	VIA1 Size X	VIA2 Size X	VIA3 Size X	VIA4 Size X	VIA5 Size X	Note
NTAIL	Nil	1	-0.6,1.0	-0.6,1.0	-0.6,1.0	-0.6,1.0	-0.6,1.0	Min via=2
CMFB	Nil	Nil	-0.6,1.0	-0.6,1.0	-0.6,1.0	-0.6,1.0	-0.6,1.0	
INPUT	Nil	Nil	-0.6,1.0	-0.6,1.0	-0.6,1.0	-0.6,1.0	-0.6,1.0	
NCAS1-NCAS2	1-C	Nil	-0.6,1.0	-0.6,1.0	-0.6,1.0	-0.6,1.0	-0.6,1.0	
NCAS2-NCAS1	1-C	Nil	-0.6,1.0	-0.6,1.0	-0.6,1.0	-0.6,1.0	-0.6,1.0	
PCAS1-PBIAS	1-2C	Nil	-0.6,1.0	-0.6,1.0	-0.6,1.0	-0.6,1.0	-0.6,1.0	
PCAS2-NCAS2	1-C/2	Nil	-0.6,1.0	-0.6,1.0	-0.6,1.0	-0.6,1.0	-0.6,1.0	
PBIAS-PCAS1	1-2C	Nil	-0.6,1.0	-0.6,1.0	-0.6,1.0	-0.6,1.0	-0.6,1.0	

For connecting Source/Drain of a transistor using MET2,

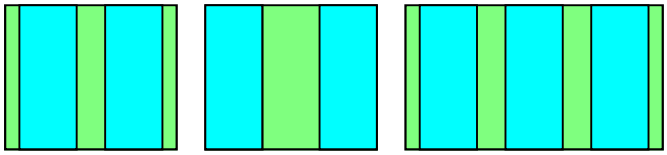
$$MET2\_WIDTH = MOS\_WIDTH\ /(noMetRow * 1.5)$$

$$MET2\_OFFSET = 0.5 * MET2\_WIDTH$$

$$OFFSET\_EDGE = 0.25 * MET2\_WIDTH$$

Case 1: Number of MET2 S/D, noMetRow =2, the routing is shown in Figure 4(a) where MOS_WIDTH large enough.

$$MET2\_WIDTH = MOS\_WIDTH/(2 * 1.5)$$
$$= MOS\_WIDTH/3.0$$



(a) (b) (c)

Figure 4: Metal MET2 routing for connecting source and drains together in a transistor; green is transistor, blue is metal 2.

Case 2: Number of MET2 S/D, noMetRow =2, the routing is shown in Figure 4(b) where MOS_WIDTH is small. If width of transistor is very small (in case of CMFB, NCAS2 and PCAS2 split) then the spacing from edge can be made zero and S/D can be routed as shown in Figure 4(b). If MET2 width is minimum and spacing required between metals is not sufficient as per DRC then this approach is implemented.

Case 3: No of MET2 S/D, noMetRow =3, the routing is shown in Figure 4(c).

$$MET2\_WIDTH = MOS\_WIDTH/(3 * 1.5)$$
$$= MOS\_WIDTH/4.5$$

Length of MET2 is equal to the difference of Y co-ordinates of first and last source/drain bounding box of pcell transistor.

For connecting S/D of multi-row-column transistor, a MET2 will be routed from top to the base of the transistor and VIA1 are placed alternatively to connect S/D. For placing via, the common area of two metals is taken and filled with maximum number of via array. There will be a Relative Object Database (ROD object), which will do this.

2.4. Routing of inter transistor source and drain

MET3 and MET4 are used for inter-source drain connection. MET5 is used for NTAIL source to AVSS connection. For gate and substrate connections MET1 is used. For connecting inter-transistor Source/Drain using MET3, MET3 width, MET3 Spacing and No of MET3 S/D strips are calculated as follows:

```
1.  MET3_WIDTH=MAX_MET3_WIDTH
2.  LENGTH_MET2=Difference (y co-ordinates of first and last
    source/drain of transistor)
3.  MET2_CENTER=Average (Lower and Upper Y co-ordinates of
    MET2 rectangle.)
4.  NO_OF_S=round (LENGTH_MET2/(3.5*MET3_WIDTH))
5.  /*Find a MET3_WIDTH, which gives Odd number of source MET3
    strips.*/
6.  while( evenp (NO_OF_S)
                MET3_WIDTH=MET3_WIDTH-0.1
                NO_OF_S=floor(LENGTH_MET2/(3.5*MET3_WIDTH))
7.  )

8.  /*Check if it is less then 3? If yes use 3 and calculate the
    MET3_WIDTH else WARN. */

9.  if(NO_OF_S<3 then
      NO_OF_S=3
          MET3_WIDTH=fixDeci(pcCellView LENGTH_MET2/10.5);
     if(MET3_WIDTH<3.0 then
                warn("PCELL ERROR!!!  MET3 WIDTH IS VERY SMALL
                FOR INPUT CONNECTION..")
10.      );if MET3_WIDTH
11. );if NO_OF_S
12. NO_OF_D=NO_OF_S-1
13. SD_OFFSET_NEW= MET3_WIDTH *3.5
14. CCOY_S = INPUT_MET_CENT-((NO_OF_S-1)/2.0)
    *SD_OFFSET_NEW
15. CCOY_D = CCOY_OK+(SD_OFFSET+MET3_WIDTH)/2.0
```

The spacing between the MET3 is 0.75 * MET3_WIDTH. For LENGTH_MET2=110 and MAX_MET3_WIDTH=10 NO_OF_S=3, MET3_WIDTH =10, MET3_OFFSET=7.5. The example routing is as shown in Figure 5-7.

For connecting inter-transistor source/drain using MET4, MET4 width is taken from user as input. MET4 Spacing and No of MET4 S/D strips are calculated as follows.

- For PBIAS-source connection, the MET4 is used for connecting its source terminal to VDD. The spacing from Y-axis is zero. Only one MET4 rectangle is to be drawn.
- For PBIAS-PCAS1 source-drain connection, the number of MET4 connections is equal to minimum of number of columns in PBIAS and PCAS1.
- For PCAS1-PCAS2 source-drain connection, the number of MET4 connections will be equal to half the number of columns in PCAS2.
- For NCAS2-NCAS1 source-drain connection, the number of MET4 connections will be equal to the number of S/D columns in NCAS1.
- For NCAS2-NCAS1 metal connections, MET2 is extended through MET4 and connected to NCAS2.
- For via placement the spacing from edges in x and y directions is fixed (e.g. 0.6um and 1.0um respectively for all type of vias) as shown in Figure 7.
- For connecting NTAIL Source to VSS using MET 5, the MET5 width is taken from user as an input for connecting its source to VSS. The spacing from Y-axis is zero. Only one MET5 rectangle is drawn.

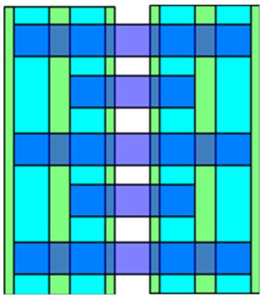


Figure 5: Routing using MET3 of two splits of a transistor.

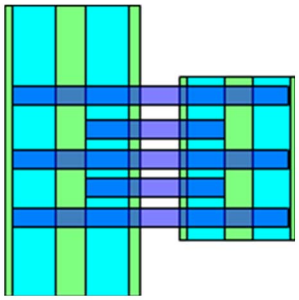


Figure 6: Inter transistor routing using MET3

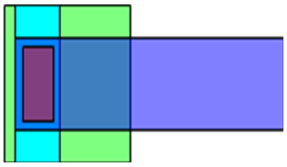


Figure 7: Via spacing from the edge of the overlapping metals, MET2 and MET3.

2.4. Gate Routing

MET1 is running horizontally between transistors to connect gates of PCAS2, NCAS2, NCAS1, INPUT CMFB and NTAIL and to other half of pcell as well as to the outside connections as shown in Figure 8. INPUT gate connection can be one or two based on the user selection. The minimum MET1 width is used for routing. User can specify the width.

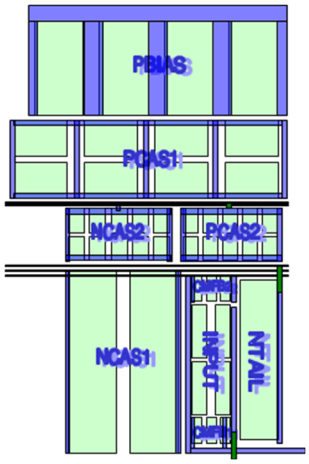


Figure 8: Inter and intra transistor gate routing using MET1

2.5. Substrate Routing

MET1 is running for substrate connections of fixed width (1um) horizontally and (of 2um) vertically. The minimum MET1 width is used for routing. User can specify these widths.

3. Results and Discussion

The proposed approach is implemented as Super pcell in Cadence's SKILL language. All the required pCell routines are loaded in ICFB at startup. In order to generate the Super pCell a designer needs to create a schematic as shown in Figure 9. The generated Super pCell is shown in Figure 10. As discussed earlier the mega pCell is created for half-cell and two half-cells are placed with R0 and MX orientation to create super pCell. The generated pCell is DRC and LVS clean. Figure 11 shows the generated super pCell in three different aspect ratios. As can be seen from the layout the NCAS1, INPUT and CMFB splits have been altered to achieve different aspect ratios. Figure 12 shows the component description parameters of the pcell. There are basic parameters such as number of row and column offsets, metal width etc. and advanced parameters which are defined by advanced variables in the variable information field. The generated pCell is fully controllable. Designers can quickly change these parameters and see the altered layout instantly and complete the optimization in couple of minutes which otherwise would have taken days. Finally pCell can be flattened to further fine tune the layout if required. This gives layout engineer full flexibility of hand drawn layout. The generated

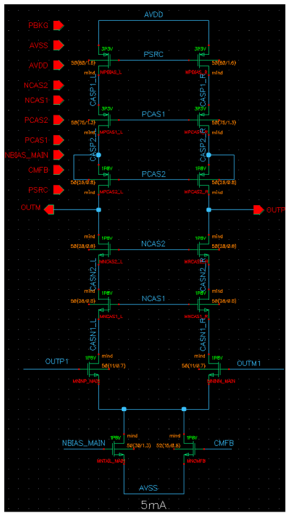


Figure 9: Schematic of triple cascade amplifier

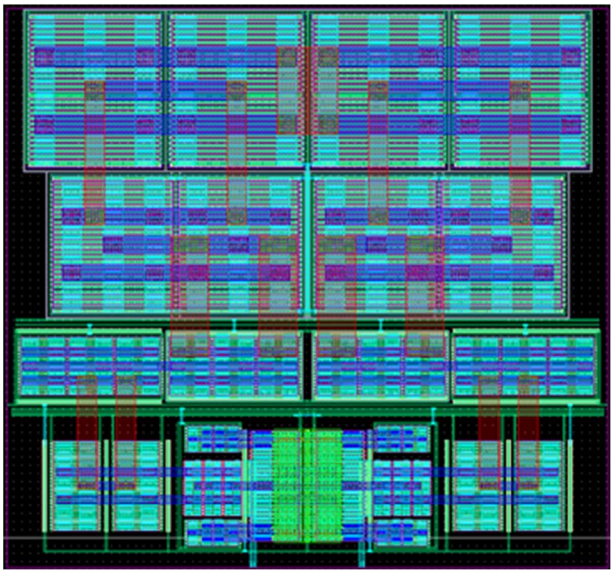


Figure 10: Triple cascade amplifier layout generated using Super pCell.

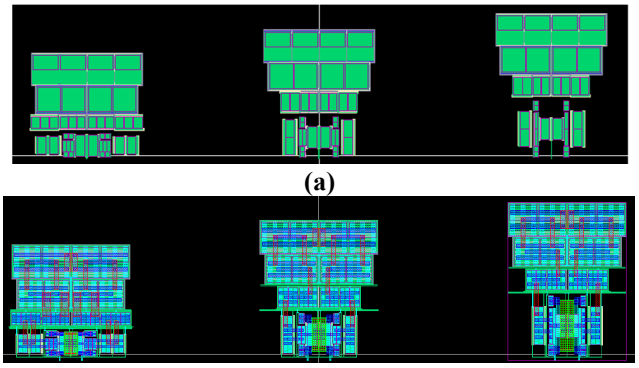


Figure 11: Generated Super pCell for three different aspect ratios: a) placement of transistor b) placement and routing.

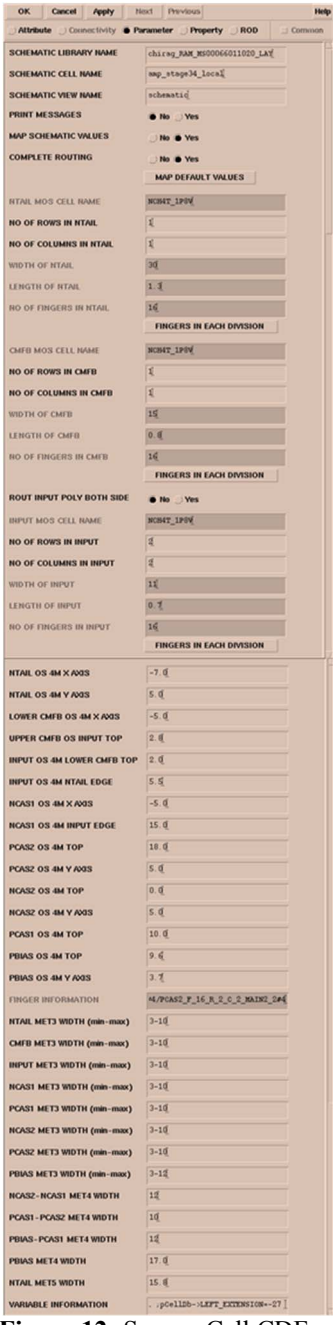


Figure 12: Super pCell CDF parameters

pCell has been extensively used by layout engineers and very well adopted by the design community due to ease of use and full flexibility. This approach has been used for smaller analog blocks such as current mirror and differential pairs, as well as IO Ring and bigger blocks such as complete LDO and found to be very effective in reducing the analog design cycle time. In this work the pCell has been hand coded. However there are automated pCell creation tool and automated frameworks available which can speed up the pCell creation time significantly [20].

4. Conclusion

In this paper a simplified methodology for complex analog module layout generation using template driven parameterized cell for analog layout automation is presented. With the proposed methodology the layout creation time is reduced from a week to few secs. The generated layout is LVS and DRC clean which increases efficiency of layout engineers. Optimization of the final amplifier layout can be done very easily within few minutes with generated pCell. The approach can be extended to other reusable analog modules. The creation of templates is knowledge intensive task and the set of generated layout is limited by available templates. Therefore the approach is limited to reusable analog modules within a design and/or across multiple designs. Automated pCell generation tools and framework can significantly reduce the pCell creation time.

5. References

[1] Prautsch, Benjamin, et al. "MESH: Explicit and flexible generation of analog arrays." Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), 2017 14th International Conference on. IEEE, 2017.

[2] J. Crossley et al. "BAG: A Designer-Oriented Integrated Framework for the Development of AMS Circuit Generators" 2013 IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD) pp. 74-81 2013.

[3] A. Graupner R. Jancke R. Wittmann "Generator Based Approach for Analog Circuit and Layout Design and Optimization" Design Automation & Test in Europe Conf. & Exhibition (DATE) pp. 1-6 2011.

[4] B. Prautsch et al. "IIP Framework: A Tool for Reuse-Centric Analog Circuit Design" 13th Int. Conf on Synthesis Modeling Analysis and Simulation Methods and Applications to Circuit Design (SMACD) June 2016.

[5] T. Reich U. Eichler K.-H. Rooch R. Buhl "Design of a 12-bit Cyclic RSD ADC Sensor Interface IC Using the Intelligent Analog IP Library" ANALOG 2013-Entwicklung von Analogschaltungen mit CAE-Methoden March 2013.

[6] R. Castro-López O. Guerra E. Roca F. V. Fernández "An Integrated Layout-Synthesis Approach for Analog ICs" IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems vol. 27 no. 7 pp. 1179-1189 July 2008.

[7] A. Unutulmaz G. Dündar F. Fernández "LDS based Tools to Ease Template Construction" Synthesis Modeling Analysis and Simulation Methods and Applications to Circuit Design (SMACD) 2012 Int. Conf. pp. 61-64 2012.

[8] R. Martins et al. "AIDA: Automated Analog IC Design Flow from Circuit Level to Layout" Proc. of the Int. Conf. on Synthesis Modeling Analysis and Simulation Methods and Applications to Circuit Design (SMA CD) 2012.

[9] H. Graeb et al. "Analog Layout Synthesis - Recent Advances in Topological Approaches" Proc. of the Conf. on Design Automation and Test in Europe 2009.

[10] R. Martins N. Lourenco N. Horta "LAYGEN II—Automatic Layout Generation of Analog Integrated Circuits" IEEE Transactions on Computer-Aided Design of In tegra ted Circuits and Systems pp. 1641-1654 2013.

[11] H. Habal H. Graeb "Constraint-Based Layout-Driven Sizing of Analog Circuits" IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems vol. 30 no. 8 pp. 1089-1102 2011.

[12] G. G. E. Gielen R. A. Rutenbar "Computer-Aided Design of Analog and Mixed-Signal Integrated Circuits" Proc. of the IEEE 88.12 pp. 1825-1854 December 2000.

[13] A. Krinke G. Jerke J. Lienig "Constraint Propagation Methods for Robust IC Design" Proc. of ZuE 2015; 8. GMM/ITG/GI-Symp. Reliability by Design pp. 1-8 2015.

[14] A. Nassaj J. Lienig G. Jerke "A New Methodology for Constraint-Driven Layout Design of Analog Circuits" Proc. 16th IEEE Int. Conf. on Electronics Circuits and Systems pp. 996-999 2009.

[15] J. Scheible J. Lienig "Automation of Analog IC Layout - Challenges and Solutions" Proc. of the 2015 Int. Symp. on Physical Design pp. 33-40 2015.

[16] D. Marolt J. Scheible G. Jerke V. Marolt "SWARM: A Self-Organization Approach for Layout Automation in Analog IC Design" Int. Journal of Electronics and Electrical Engineering (IJEEE) vol. 4 no. 5 pp. 374-385 2016.

[17] B. Prautsch U. Eichler T. Reich J. Lienig "Explicit Feature and Edge Insertion for Improved Analog Layout Generators in Advanced Semiconductor Technologies" Proc. of ANALOG 2016 pp. 22-27 September 2016.

[18] Lomeli-Illescas, Ismael, et al. "Synthesis tool for automatic layout generation of analog structures." ANDESCON, 2016 IEEE. IEEE, 2016.

[19] Cadence SKILL Users Guide

[20] Cadence pCell Users Guide

[21] https://www.cadence.com/content/cadence-www/global/en_US/home/training/all-courses/84422.html

[22] https://sus.ziti.uni-heidelberg.de/Lehre/WS1718_VLSI/VLSI_Fischer_05_Skill.pptx.pdf