Practical Placement and Routing Techniques for Analog Circuit Designs

Linfu Xiao, Evangeline F.Y. Young
Department of CSE
The Chinese University of Hong Kong
Shatin, N.T. Hong Kong
e-mail: {lfxiao, fyyoung}@cse.cuhk.edu.hk

Xiaoyong He, K.P. Pun
Department of EE
The Chinese University of Hong Kong
Shatin, N.T. Hong Kong
e-mail: {xyhe, kppun}@ee.cuhk.edu.hk

***Abstract*—[1]In this paper, we will present an effective layout method for analog circuits. We consider symmetry constraint, common centroid constraint, device merging and device clustering during the placement step. Symmetric routing will then be performed. In order to have successful routing, we will perform analog-based routability-driven adjustment during the placement process, taking into account for analog circuits that wires are not preferred to be layout on top of active devices. All these concepts were put together in our tool. Experimental results show that we can generate quality analog layout within minutes of time that passes the design rule check, layout-schematic verification and the simulation results are comparable with those of manual design, while a manual design will take a designer a couple of days to generate.**

I. Introduction

The integration of high-performance analog and digital circuits leads to an increasing need of new tools compatible for both the digital and analog parts. Unfortunately, the low acceptance of CAD tools in the analog domain presents a serious bottleneck to a fast realization of mixed-signal systems. Due to a higher sensitivity of the electrical performance to layout details, analog designs are much more complicated than digital ones. Process and temperature variations can introduce severe mismatches in devices that are designed to behave identically. These undesirable effects can be alleviated by a symmetric layout. Matching and symmetry in placement and routing of analog circuits are thus very important.

Layout design of analog circuits is an error prone and time-consuming process. Some devices need to be placed in close proximity and symmetrically with respect to an axis or to a center point. This can reduce the effect of parasitic mismatches which, if not properly controlled, will cause significant degradation of circuit performance.

A. Previous Works

The analog placement problem has been studied extensively. In 1-D symmetric placement, cells are required to be placed symmetrically with respect to a horizontal or a vertical axis. In 2-D symmetric (or common centroid) placement, cells are required to be placed symmetrically with respect to a single point. Simulated annealing is often used to solve this symmetric placement problem. There are two categories of works, one using an absolute representation of the placement and one using a topological representation. The works by [16], [17], [18] uses absolute representation and each module is located by an absolute coordinates. In absolute representation, any arbitrary constraint can be formulated directly and every possible placement can be described. However, the solution space is infinitely large and contains many infeasible ones with overlapping between modules.

[1]The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK418908).

In topological representations, the relative positionings between the modules are used to described a placement. This leads to a much smaller solution space without any infeasible overlapping solution. Many topological representations have been used and extended to solve this analog placement problem, including sequence pairs [3], [10], [7], [11], [12], segment tree [19], B*-tree [8], [5], O-tree [4] and TCG [6]. Lin [8] devised a hierarchical module clustering-based method based on the B*-tree representation to handle matching, 1-D symmetry, and proximity constraints. For the common centroid constraint, Ma and Young [1] proposed a method based on a center-based corner block list (C-CBL) representation. Strasser [9] proposed a deterministic placement algorithm based on B*-tree using hierarchically bounded enumeration and enhanced shape functions to handle 1-D and common centroid constraints.

B. Our Contributions

Although the problem of symmetric placement for analog circuits has been extensively studied, most of the previous works focused only on the placement process. In this work, we aim at producing a tool that, given an analog circuit schematic, can generate a complete and quality layout passing the design rule check, the layout-schematic verification and the performance is verified through simulation. Besides common centroid and 1-D symmetry placement, we handle several other important features in analog circuit layout, e.g., device merging (sharing geometry between devices to reduce area), device clustering (clustering devices of the same kind for better matching and routing properties). Routing is performed after placement using a modified maze router, which can take care of symmetric wires. Last but not least, we propose a practical approach to take congestion into account during the placement process for analog circuits since wires are preferred not to be layout above active area of devices, and we thus need to leave enough routing spaces between the devices in order to be able to complete routing successfully at the end. In the experiments, we compare our method in handling 1-D symmetry constraints in placement with previous works and show that our placer can produce good symmetric layouts efficiently. We then demonstrate the whole flow of device extraction, placement, routing and verification using two realistic analog circuits. [2] Results show that our tool can automatically generate quality layout for analog circuits very efficiently, while manual design will take a designer a couple of days to generate.

The remainder of this paper is organized as follows. Section 2 gives the system overview of our approach. Section 3 presents the first step of our method, device layout retrieval. The analog placement process will be presented in Section 4. Section 5 describes the analog symmetric router. Experimental

[2]We will make these benchmarks publicly available.

results including the design rule check, layout-schematic verification and simulation will be shown in Section 6, followed by a conclusion in Section 7.

II. An Overview of Our Approach

The flow of our system can be summarized in Fig. 1. Taking as input a netlist file of devices, we first retrieve all the basic device layouts by looking at the process layout library supplied by the IC foundry. A placement that considers symmetry constraint, common centroid (CC) constraint, device merging, device clustering and routability will then be generated by our placer and based on that, symmetric routing will be performed. Finally, verification including design rule check (DRC), layout-schematic (LVS) check and simulation will done. A verified layout of the input analog circuit will be output at the end.

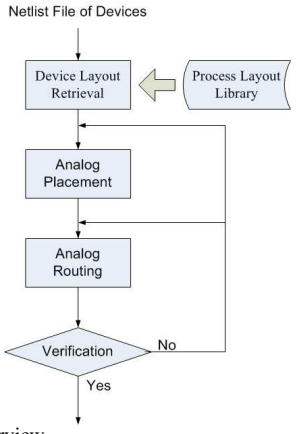

Fig. 1. System Overview

III. Device Layout Retrieval

This step makes use of the Cadence analog IC design tool icfb Virtuoso platform. Once an IC designer designed a circuit using the schematic view method, we can get the netlist file exported by Virtuoso. We will then retrieve the layouts for all the devices used in the input analog circuit from a process layout library obtained from the Virtuoso Layout XL tool in the GDSII format. There are several types of basic devices: resistor, capacitor, transistor and diode, etc. All the basic devices are parameterized cells (P-cell), whose layouts are already stored in the process layout library. Once all the parameters of a device are obtained by parsing the netlist file, we can obtain its layout from the layout library.

A. Extraction of Device Pins

With the layout of each device, we can obtain the pin polygons according to the process technology file. For example, for PMOS, the polygons on metal layer 1 are the drain or source pins, the polygon on the poly layer is the gate pin and the polygon in the nwell/pwell layer is the bulk pin. We will then pick the center of each polygon as the pin position.

IV. Analog Placement

For the analog placement problem, we are given a netlist of devices and their pin positions, some devices may be specified as belonging to one symmetry group and are required to be placed symmetrically with respect to a single point (common centroid constraint) or to a horizontal or a vertical axis (1-D symmetry constraint). The objective is to generate a non-overlap placement of the devices with good analog routability such that all the symmetry requirements are satisfied. We use simulated annealing with sequence pair in our placement engine. To further reduce area, wire length and parasitic effect, dynamic device merging and device clustering are considered. In device merging, the geometric area of some devices is shared to improve the layout density and performance. In device clustering we put some devices of the same kind in close proximity to reduce mismatch errors. Since the active area of devices should be considered as blockage in analog routing for better performance, we derived a congestion-driven placement scheme to ensure good analog routability. We will discuss all these issues in the following sections.

A. Handling Common Centroid and Symmetry Constraints

In analog layout, symmetry is an important requirement to reduce mismatches. In some cases when the devices are small and the requirement in accuracy is not that stringent, a common centroid constraint can be relaxed to a 1-D symmetry constraint with devices placed in close proximity. Both common centroid and 1-D symmetry constraints are thus important to analog placement. The paper [13] proposed a method to handle both types of constraints in analog placement using simulated annealing with sequence pairs as the representation. Based on this placement engine, other analog related features like device merging, device clustering and congestion estimation are developed.

Our placement engine is simulated annealing based using sequence pair as the representation. A sequence-pair (SP) [2], describing a general placement of n blocks, is an ordered pair (α,β), where α and β are permutations of the block names. We use α_i to denote the block occupying the i^{th} position in sequence α and use α_A^{-1} to denote the position of block A in the α sequence.

Some symmetric feasible conditions on sequence pair are identified in [13] that can cover completely without redundancy the set of all feasible placements satisfying the common centroid or 1-D symmetry constraints. The searching process can thus be done effectively in a complete and non-redundant solution space. It was shown in [13] that the feasible condition for common centroid constraint of a symmetry group g in a sequence pair (α,β) is that

$$\alpha_g = sym(rev(\alpha_g)) \tag{1}$$

$$\beta_g = sym(rev(\beta_g)) \tag{2}$$

where α_g and β_g are the extracted sub-sequence pair of g containing only the devices in group g (e.g., if the sequence pair is $(badec, aebdc)$ and group g has device a, b and c, the sub-sequence pair of g will be (bac, abc)), the operation $rev(s)$ is reversing the string s and the operation $sym(s)$ is replacing all the blocks in string s by their symmetric counterparts. Similarly, for a group g with 1-D symmetry constraint, the symmetric feasible condition is:

$$\alpha_g = sym(rev(\beta_g)) \tag{3}$$

B. Device Merging

Device merging, also called geometry sharing, is a common technique used in analog layout design. In the following, we

call the devices with connected bulk pins a *merging component*. Devices in one merging component may be merged in several groups and we call the devices (must be in the same component) that share their geometries a *merging group*. Devices in a merging group will share their geometries to reduce the placement area, wire length and parasitic effect. Fig. 2 shows an example on how device merging works. The minimal distance between two resistors is specified by the layout physical rules.

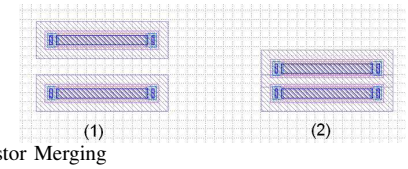

Fig. 2. Resistor Merging

1) Merge Feasible Condition: Devices in one merging group g should align either horizontally or vertically. To achieve this in the sequence pair representation, firstly, the sub-sequence pair of a merging group g should appear contiguously in the original sequence pair. Besides, the sub-sequence pair (α_g, β_g) of a merging group g must satisfy the following relationship:

$$\begin{array}{ll} \alpha_g = \beta_g, \text{ or} & \text{(align horizontally)} \\ \alpha_g = rev(\beta_g) & \text{(align vertically)} \end{array} \quad (4)$$

The above two conditions make up the *merge feasible condition* for a merging group.

2) Dynamic Merging: In our placement method, merging between devices are determined dynamically during the annealing process. We first do a pre-process to generate all the merging components before the annealing process. To trade off between area and wire-length, devices in one merging component may merge as several groups, not necessarily as one big group. An example is shown in Fig. 3. There are eight resistors in the merging component, we can either merge all of them as one group (left part), or merge as two groups (right part).

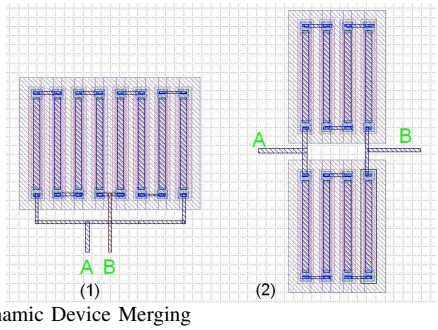

Fig. 3. Dynamic Device Merging

We consider dynamic merging in the annealing process by merging and separating devices in the random moves. We use a graph G to represent the merging groups of a merging component. The nodes in G correspond to the devices, and there is an edge between two nodes if the two devices are merged in the current solution. Each connected component in G corresponds to a merging group. During the annealing moves, we will randomly merge two connected components or separate one connected component into two in the graph G. We will then map the relationships of the nodes in G back to the sequence pair accordingly.

C. Device clustering

We also consider device clustering for better circuit performance. Device clustering means that some devices of the same kind are placed in close proximity to reduce mismatch errors and parasitic effects. In our approach, we apply different clustering techniques for different types of devices.

1) Handling of Capacitors: Capacitors with the same size as well as in parallel can be placed as one big *device array* of size $m \times n$. In the annealing process, these m and n can be changed to consider putting the capacitors together in different ways.

2) Handling of Transistors: In practice, about 90% of NMOS and PMOS have their source pins connected to the ground or to the V_{dd}. In order to achieve stable power supply and reduce the IR-drop effect, all the NMOS (PMOS) will be placed close to each other. We also require them to be placed in the same orientation to optimize routing to the ground and to the V_{dd}. Therefore, we consider NMOS and PMOS as two super-blocks and require them to be contiguous in the sequence pair representation.

D. Congestion Aware Placement

To reduce parasitic capacitance and cross-talk effect, the routing spaces above the active area of the devices are often avoided (considered as obstacles) in the routing step. A compact placement is thus not practical in analog design. Blockage aware congestion estimation during the analog placement step is essential such that enough spaces will be reserved between the devices for laying out of the wires. None of the previous works has considered such blockage aware congestion-driven placement for analog designs. We propose a method to handle this problem as follows.

After we obtain a compact placement by realizing a candidate sequence pair solution, a coordinate adjustment step will be done, as shown in algorithm 1. Firstly, we will divide the whole placement region into a $n \times n$ mesh (n is set to 40 in the experiments) and calculate the vertical and horizontal wire capacities for each room according to the minimum wire width. We will then estimate the congestion for each room and expand it based on the congestion map. The area and HPWL of this expanded placement will be used in the computation of the cost function for the current candidate solution, which is more accurate than the original one.

E. Types of Moves

At the highest level of a candidate sequence pair, there will be NMOS clusters, PMOS clusters, capacitor arrays, resistor clusters and other individual blocks like diodes, etc. For each NMOS/PMOS or resistor cluster, devices with connected bulk pins may form merging groups. We employ the following set of moves to perturb a candidate solution.

- Swapping on the highest level sequence pair - Randomly pick two items X and Y which can be clusters, device arrays or individual blocks (not belonging to any cluster or device array), swap them in the α (β) sequence.
- Swapping within a cluster X -
  - Swap two individual blocks in X.
  - Swap two merging groups in X.
  - Swap a block and a merging group in X.
- Change the device order in a merging group - Randomly pick two blocks in a merging group, swap them in both sequences.

Algorithm 1 Congestion Aware Placement Adjustment

1: Divide the layout into a $n \times n$ mesh. Assume that the room width and height are w and h respectively.
2: Assume that the minimum wire width is z, $c_v = h/z$; $c_h = w/z$
3: Calculate the routable space percentage X for each room.
4: // Congestion estimation:
5: **for** each net **do**
6: Calculate the horizontal and vertical congestion measures [15] for each room at (i, j). Add them to $cong_v[i][j], cong_h[i][j]$.
7: **end for**
8: // Expand grid:
9: **for** each room (i, j) **do**
10: $\delta = cong_v[i][j] - c_v \times X[i][j]$.

$$expand_h[i][j] = \begin{cases} 0 & : \delta < 0 \\ \delta & : otherwise \end{cases} \quad (5)$$

11: $\delta = cong_h[i][j] - c_h \times X[i][j]$.

$$expand_v[i][j] = \begin{cases} 0 & : \delta < 0 \\ \delta & : otherwise \end{cases} \quad (6)$$

12: **end for**
13: Take the maximum value for each column and row.
$expand_v[j] = \max_{i=1}^{n} expand_v[i][j] \ \forall j = 1 \ldots n.$
$expand_h[i] = \max_{j=1}^{n} expand_h[i][j] \ \forall i = 1 \ldots n.$
14: Re-calculate the position for each room.
15: Move the center of each device to the same relative position of the same room.

- Change the orientation of a block - There are 4 orientations for each device (north, south, west, east). Randomly pick one block and change its orientation.
- Change the aspect ratio of a device array - Randomly pick a $m \times n$ device array, choose randomly another row and column number.
- Dynamic device merging - Details refer to section 4.2.2.

F. Annealing Schedule and Cost Function

In our annealing engine, the initial temperature is set to 10^6 and the temperature will drop at a constant rate. At each temperature, k iterations are performed, where k is proportional to the number of blocks. We use the cost function $cost(F) = area(F) + \lambda * wire(F)$ to evaluate a packing F, where $area(F)$ is the area of F and $wire(F)$ is the total wire length estimated by the half perimeter method. Before the annealing process, a random walk with K moves ($K = 1000$ in the experiments) will be performed to determine the value of the parameter λ in the cost function, where we try to balance the relative weighting between area and wire length.

V. Analog Routing

In routing, we apply a grid-based routing scheme, using unreserved layer model with two metal layers. We will route most of the nets using wires of minimum width except those critical ones with wider widths. The width (and height) of each grid element is thus the sum of the wire width and the minimum wire spacing. Some nets are required to be routed symmetrically to reduce mismatches. The active areas of all the devices are considered as routing blockages.

We will in advance sort all the nets and then route them one by one using a multi-pin maze routing engine [14]. The following criteria are considered in the net routing order with decreasing priority:
(1) symmetry nets, (2) nets with less bounding box areas, and (3) nets with less number of pins. If overflow occurs at the end, rip-up-and-reroute will be performed until there is no more overflow.

A. Symmetric Routing

To route two nets symmetrically, we first route one of them, then the route is mirrored to produce the route of its counterpart. In order to guarantee a successful *mirroring step*, we will take the union of all the blockages on both sides to produce the first route which will then be mirrored to give the second route.

B. Layer Assignment

When the grid-based multi-pin maze routing engine [14] is applied, we assume a reserved layer model with two metal layers in the H-V or V-H format. After this routing step, we will do layer assignment on the routing result to minimize the number of vias used and to reduce crosstalk.

There are previous works on this layer assignment problem on two metal layers to minimize the number of vias used optimally. This can be done by constructing a *conflict-continuation graph* [20]. We implemented this layer assignment approach into our router and extended it to consider vias of degree four and the crosstalk effect between wires. This extension can be done by adding a continuation edge between two wire segments which are close to each other and run in parallel for a long enough distance.

VI. Experimental Results

Our analog layout tool was implemented in C++ and all the experiments were performed on an Intel(R) Core(TM) 2 Duo CPU 2.20GHz linux workstation with 4 GB memory. We have done two sets of experiments. In the first set, which is a pure placement problem, we compared our approach with the most updated previous works [3], [5], [10], [7], [8], [11] and [9] handling 1-D symmetry constraint, using two industrial designs, biasynth_2p4g and lnamixbias_2p4g. The results are shown in Table I[3]. In this table, the third column on "1-D Groups" shows the 1-D symmetry group information. We can see that our approach is efficient, and the area usage and running time of our work are among the best.[4]

In the second set of experiments, we tested our tool on its effectiveness in going through the whole flow of device extraction, placement, routing and finally generating a layout passing the DRC and LVS verification. Furthermore, we will show the quality of our result by performing a post-layout simulation. We use the UMC 0.13*um* process technology for the analog design environment.

We perform the second set of experiments on two Operational Transconductance Amplifiers (OTA) from industry, and ultra-low voltage supply ($0.5V$) is used. Table II lists our results. In this table, columns $2-5$ show the circuit information. Columns $8-9$ show the wire length and via number of the resultant layouts. Dead spaces are shown in the last two columns of table III comparing with the manual designs. We can see that our tool can generate a reasonable layout very efficiently, while it may take an experience layout engineer a couple of days in manual design. Fig. 4 shows the resultant layouts for OTA1.

[3]Results are obtained from [9]

[4]It is hard to compare directly the running time, since different machines have been used.

TABLE I
COMPARISON OF AREA USAGE AND RUNTIME FOR DIFFERENT APPROACHES ON 1-D SYMMETRY PLACEMENT

Data Set	Block No.	1-D Groups	SP [3]		ST [5]		SP+LP [10]		SPwD [7]		HB*-tree [8]		SP+JPQ [11]		B*-tree [9]		Our Work	
			area	time	area	time	area	time	area	time	area	time	area	time	area	time	area	time
biasynth_2p4g	65	8,12,5	114.9	780	114.9	246	106.4	403	118.5	134	104.7	22	N/A	N/A	104.9	337	104.8	13.5
lnamixbias_2p4g	110	16,6,6,12,4	110.4	2824	109.4	726	108.6	3252	113.5	227	105.7	43	109	480	107.7	387	104.9	53

All running times are measured in seconds, and all area usages are measured in percentage of the total module area. SP, ST and SP+JPQ are run on Sun Blade 100 500MHz, SP+LP, SPwD, HB*-tree and B*-tree are run on Pentium4 3.2GHz.

TABLE II
RESULTS OF THE SECOND EXPERIMENT ON LAYOUT GENERATION

Data Set	Device Number					Device Area (um^2)	Wire Length (um)	Via Number	Running Time (s)
	Capacitor	Resistor	PMOS	NMOS	Total				
ota1	20	17	12	16	65	17057.8	2367.2	274	79
ota2	16	19	12	16	63	15094.7	2108.8	264	65

TABLE III
COMPARISON OF SIMULATION RESULTS OF SCHEMATIC, MANUAL LAYOUT AND OUR AUTOMATED LAYOUT

Data Set	DC Gain (DB)			Unity Gain Bandwidth (MHz)			Phase Margin			Dead Space(%)		CMRR(1KHz)		
	Schematic	Manual	Ours	Schematic	Manual	Ours	Schematic	Manual	Ours	Manual	Ours	Schematic	Manual	Ours
OTA1	57.3	55.5	56.2	70.6	69.0	69.8	67°	64.4°	64.5°	38.4	39.0	126.7	82.8	93.0
OTA2	57.8	57.4	57.6	65.4	64.5	64.2	67.3°	65.8°	64.5°	35.8	40.9	121.9	104.5	70.1

A. DRC, LVS and Post-layout Simulation

After we generate the layout automatically, the Assura tools (integrated in the Cadence Virtuoso platform) are used for the design rule check (DRC) and the layout-schematic verification (LVS). Experimental results show that our layouts can all pass both the DRC and LVS verifications.

In order to show the quality of our generated layouts, we run a post-layout simulation for each circuit. First, we perform a post-layout extraction (PEX) to obtain a new netlist from the layout with extracted parasitic RC (RCX) by using the Assura tools. We then use Analog Artist on the Cadence analog design platform to do simulation. We run simulation on the new netlist, to measure various performance parameters of the operational transconductance amplifier, such as DC gain, unity gain bandwidth , phase margin and $1KHz$ CMRR. Table III shows the simulation results. The columns of 'Schematic', 'Manual' and 'Ours' show the results of the schematic simulations, the manual layouts drawn by an experienced analog designer and our layouts respectively. We can see that the quality of our automated layouts is comparable to the manual ones.

VII. CONCLUSION

In this paper, we presented an efficient layout method for analog circuits. Our analog placer considers symmetry constraint, common centroid constraint, device merging and device clustering in placement. Furthermore, we point out that routability-driven placement is essential for analog circuits. We perform an analog-based routability-driven adjustment during the placement process. We also presented an efficient symmetry router. Experimental results show that we can generate within minutes quality layouts passing the design rule check, the layout-schematic verification and with performance comparable with manual designs according to post-layout simulation.

REFERENCES

[1] Qiang Ma *et al.* Analog Placement with Common Centroid Constraints. *Proceedings of the International Conference on Computer-Aided Design*, 2007

[2] H. Murata *et al.* Rectangle-Packing-Based Module Placement. *Proceedings IEEE International Conference on Computer-Aided Design*, 1995.

[3] F. Balasa *et al.* Symmetry within the Sequence-Pair Representation in the Context of Placement for Analog Design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2000.

[4] Y.-X. Pang *et al.* Block Placement with Symmetry Constraints based on the O-tree Nonslicing Representation. *Proceedings of the 37th ACM/IEEE Design Automation Conference*, pages 464-467, 2000.

[5] F. Balasa *et al.* On the Exploration of the Solution Space in Analog Placement with Symmetry Constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(2):177-191, 2004.

[6] J.M. Lin *et al.* Placement with symmetry constraints for analog layout design using TCG-S. *IEEE Asia South Pacific Design Automation Conference*, 2005.

[7] Y.-C. Tam *et al.* Analog Placement with Symmetry and Other Placement Constraints. *Proceedings of the International Conference on Computer-Aided Design*, 2006.

[8] P.-H. Lin *et al.* Analog Placement Based on Novel Symmetry-Island Formulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2009.

[9] M. Strasser *et al.* Deterministic Analog Circuit Placement using Hierarchically Bounded Enumeration and Enhanced Shape Functions. *Proceedings IEEE International Conference on Computer-Aided Design*, 2008.

[10] S. Kouda *et al.* Linear Programming-based Cell Placement with Symmetry Constraints for Analog IC Layout. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(4):659-668, 2007.

[11] K. Krishnamoorthy *et al.* Topological placement with multiple symmetry groups of devices for analog layout design. *IEEE International Symposium on Circuits and Systems*, 2007.

[12] Q. Dong *et al.* Constraint-free Analog Placement with Topological Symmetry Structure. *IEEE Asia South Pacific Design Automation Conference*, 2008

[13] L.-F. Xiao and Evangeline F.Y. Young. Analog Placement with Common Centroid and 1-D Symmetry Constraints. *IEEE Asia South Pacific Design Automation Conference*,2009

[14] L. Li and Evangeline F.Y. Young Obstacle-avoiding Rectilinear Steiner Tree Construction. *Proceedings IEEE International Conference on Computer-Aided Design*, 2008.

[15] C.-W. Sham *et al.* Congestion Prediction in Early Stages. *International Workshop on System-Level Interconnect Prediction*, 2005.

[16] John M. Cohn *et al.* KOAN/ANAGRAM II: New Tools for Device-level Analog Placement and Routing. *IEEE Journal of Solid-States Circuits SC*, 26(3):330-342, 1991.

[17] Koen Lampaert *et al.* A Performance-driven Placement Tool for Analog Integrated Circuits. *IEEE Journal of Solid-States Circuits SC*, 30(7):773-780, 1995.

[18] Enrico Malavasi *et al.* Automation of IC Layout with Analog Constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1996.

[19] F. Balasa *et al.* Efficient Solution Space Exploration based on Segment Trees in Analog Placement with Symmetry Constraints. *Proceedings IEEE International Conference on Computer-Aided Design*, pages 497-502, 2002.

[20] R.Y. Pinter. Optimal layer assignment for interconnect, *Journal on VLSI Comput. Syst.* vol. CAS-30, pp. 284-299, 1983.

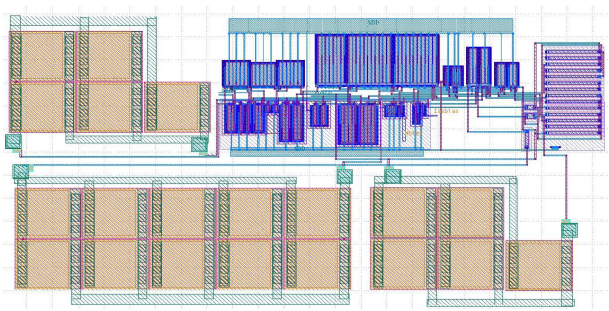

Fig. 4. Layout of OTA1