IPRAIL – Intellectual Property Reuse-based Analog IC Layout Automation

Nuttorn Jangkrajarng, Sambuddha Bhattacharya, Roy Hartono, and C.-J. Richard Shi

{njangkra,sbb,rhartono,cjshi}@ee.washington.edu

Mixed-Signal CAD Research Laboratory,
Department of Electrical Engineering, University of Washington
Seattle, WA 98195-2500, USA

UWEE Technical Report
Number UWEETR-2003-0004
March 17, 2003.

Department of Electrical Engineering
University of Washington
Box 352500
Seattle, Washington 98195-2500
PHN: (206) 543-2150
FAX: (206) 543-3842
URL: http://www.ee.washington.edu

IPRAIL – Intellectual Property Reuse-based Analog IC Layout Automation*

Nuttorn Jangkrajarng, Sambuddha Bhattacharya, Roy Hartono,
and C.-J. Richard Shi†
{njangkra,sbb,rhartono,cjshi}@ee.washington.edu

Mixed-Signal CAD Research Laboratory,
Department of Electrical Engineering, University of Washington
Seattle, WA 98195, USA

University of Washington, Dept. of EE, UWEETR-2003-0004
March 17, 2003.

ABSTRACT

This paper presents a computer-aided design tool, IPRAIL, which automatically retargets existing analog layouts for technology migration and new specifications. The reuse-based methodology adopted in IPRAIL utilizes expert designer knowledge embedded in analog layouts. IPRAIL automatically extracts analog circuit and layout intellectual properties as templates, incorporates new technology design rules and device sizes, and generates fully functional layouts. This is illustrated by retargeting two practical operational amplifier layouts from the TSMC 0.25um CMOS process to the TSMC 0.18um CMOS process. While manual re-design is known to take days to weeks, IPRAIL only takes minutes and achieves comparable circuit performances.

Keywords: Analog Integrated Circuit Design; Analog Layout Automation; Layout Symmetry; Analog Synthesis and Optimization.

---

1. INTRODUCTION

With the increasing demand for smaller, cheaper, more portable electronics in wireless communication and consumer electronics, semiconductor industry is moving towards mixed-signal systems-on-chips. Multiple functionalities such as digital, analog, and even radio frequency (RF), which used to reside on many different chips, are being converged into one or a few chips. Driven also by the need to be more powerful,

---

* This research has been supported in part by National Science Foundation (NSF) ITR Program under Grant No. 9985507 and in part by Conexant Systems.

† Corresponding author: Tel.: +1-206-2215291; fax: +1-206-5433842.
E-mail address: cjshi@ee.washington.edu.

semiconductor manufacturers continue to innovate technologies towards smaller and smaller *transistor feature sizes* (for example from 0.25um to 0.18um to 0.13um). As a result, there is an increasing need in re-designing functioning mixed-signal layouts for new technology processes.

Due to the differences in technology process properties, migrating an available layout to a new process requires an overall re-design and re-creation of a new layout. In order to speed up the design process, design engineers can utilize available *computer-aided-design* (CAD) tools to mitigate part of the jobs. For digital layout, designers can employ the scalable cell libraries and the readily available automatic place and route tools to the existing high-level VHDL or Verilog design, in order to generate a target layout. In contrary, analog designers do not have a comparable ability, which means they have to go through a full time-consuming cycle of redesigning, testing and drawing layouts. Therefore, an automated re-layout tool for analog circuits will significantly accelerate the mixed-signal circuit technology migration.

In this paper, we present a CAD tool, called IPRAIL (***I***ntellectual ***P***roperty ***R***euse-based ***A***nalog ***I***C ***L***ayout), which automatically retargets an existing analog layout to modestly new processes. The methodology we propose here is based on the '*recycling*' scheme. IPRAIL uses an already fined-tuned input layout to automatically create a *structural template*, and then imposes new device sizes and new technology process design rules on the template. From this, IPRAIL generates an output layout that satisfies all the design rules while preserving all the analog layout intellectual properties such as device/wiring alignment, matching and symmetry. IPRAIL also preserves all unique aspects of the input layout intended by designers. Some preliminary results of this work are presented in [1].

This paper is structured as follows. Section 2 exhibits issues and previous work in analog layout automation. Section 3 illustrates the proposed IPRAIL methodology. Section 4 explains the process of layout template extraction from an existing layout. Section 5 describes automatic layout generation from an extracted layout template. Section 6 presents the experimental results of IPRAIL. Section 7 points out the limitations of IPRAIL. Section 8 concludes the paper.

2. BACKGROUND

2.1 Issues in Analog Layout Automation

The strong impact of layout geometry on circuit performance makes analog layout design a very complicated task. Device matching and symmetry, parasitics, current density in interconnects, thermal, and substrate effects are of utmost importance in high performance analog circuits [2,3]. Overcoming these challenges is essential to the success of analog layout automation. The important layout issues are briefly discussed below.

2.1.1 Matching and Symmetry

Transistors designed to behave identically may exhibit finite mismatch due to asymmetry in their layout structures or locations. The asymmetry in transistor layouts is caused mainly due to the differences in channel orientations and surrounding environment of the two transistors [2]. Two transistors are deemed symmetric if their layouts are geometric mirror images of each other. For large or stacked transistors, layouts drawn by maintaining simple geometric mirroring may not establish acceptable matching due to spatial variations in process parameters like oxide thickness, mobility etc. In such cases, *common-centroid* configurations are often employed to cancel out the mismatches introduced due to process gradients.

Transistor mismatch can drastically affect analog circuit performance leading to DC offsets, finite even-order distortion and lower common-mode rejection [4]. Ensuring layout symmetry, between transistors identical by design, demands significant designer effort.

2.1.2 Floorplanning and Device Locations

Circuit performance may be significantly affected by the exact positioning of certain devices and blocks with respect to the rest of the layout. Thermal and substrate effects, together with variations in lithographic processes, demand careful floorplanning of sensitive devices and blocks.

2.1.3. Wiring Considerations

Wiring parasitics, cross-talk and coupling can severely affect sensitive signal nets. Conservative layout styles maintaining wire-spacing and wire-shielding are often employed by expert layout designers for proper functioning of analog circuits. Wire-sizing for maintaining prescribed current density and preventing electromigration are of utmost importance in analog layout design.

2.2 Previous Work on Analog Layout Automation

2.2.1. Procedural Module Generation

The earliest approaches to analog layout automation belong to the class of procedural module generation. These schemes usually employ a designer-constructed geometric template that specifies all device-to-device and device-to-wiring spatial relationships. The template generation is accomplished either through a procedural language [5,6] or a graphical user interface [7]. The actual layout generation involves filling up the template with correct device and wire sizes. The drawbacks of these methods are in their limited flexibility and high cost of template generation. Recently, [8] proposed a template-based module generation method that attempts to palliate the

flexibility issue by extensive hierarchical templating. Unfortunately, the effort required for construction of such templates exceeds the actual manual creation of custom layouts by almost an order of magnitude.

2.2.2. Macro-cell Placement and Routing

These approaches [9-12] inherit the basic design methodologies of the digital CAD world and adapt them to analog layout automation by incorporating performance-based optimization. Devices are treated as flexible blocks and may be reshaped and reoriented using slicing-tree floorplans prior to their automatic placement and routing. The layout generation proceeds without any intervention from the layout designer. While these methods are very general in principle, they require extensive computation and, more importantly, fail to incorporate the expertise of analog layout designers into the flow. Unfortunately, the lack of designer input into the automation often results in performance degradation and forestalls the acceptance of these methods in the design community.

3. PROPOSED AUTOMATIC ANALOG LAYOUT RETARGETING METHODOLOGY

The incorporation of designer experience is a major factor to the acceptance of analog layout automation. Therefore, IPRAIL draws its inspiration from the procedural module generation methods. In contrast to [8], IPRAIL facilitates layout reuse by automatically generating the templates from an existing analog layout drawn by the experienced designer. Our objective of reusing an existing layout for new process and/or specifications is accomplished by direct extraction of the knowledge embedded in the layout.

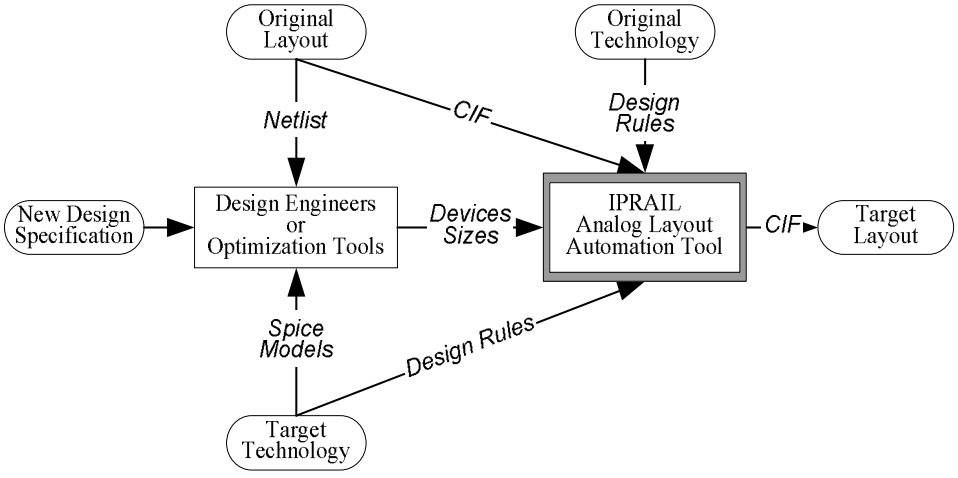


Figure 1: Analog layout retargeting methodology using IPRAIL

As illustrated in Figure 1, the *original layout* and its technology information are first fed into IPRAIL. The device sizes under the new specifications are obtained either

by manual simulations or from an analog circuit synthesis tool. First, IPRAIL converts the original layout into a *resizable symbolic template*. It then generates the new layout, henceforth called *target layout*, by imposing the target process design rules and new device sizes as constraints on the symbolic template. The entire process of automatic creation of symbolic template and generation of target layout takes a few minutes of CPU time.

The IPRAIL tool-suite consists of two main components: the *layout template extractor* and the *layout generator*. Figure 2 illustrates the internal structure and interface of IPRAIL in greater detail. First, the template extractor scans in the original layout in *Caltech Intermediate Format* (CIF) [13]. In CIF, a layout is expressed in ASCII format, which describes two-dimensional shapes on each layer based on their coordinates. The technology process design rules associated with the input layout are obtained from the Cadence environment [14].

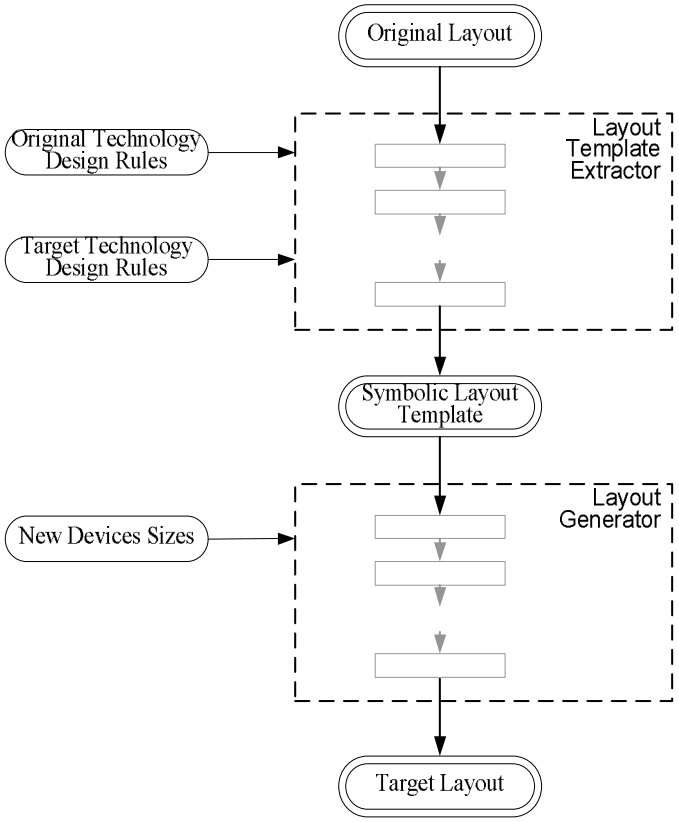


Figure 2: Overview of IPRAIL structure and flow. Details of the Layout Template Extractor and the Layout Generator are illustrated in Figure 3 and Figure 12.

The layout template extractor identifies the active and passive devices, detects device matching and symmetry, and extracts device connectivity and net-topology from the original layout. Based on the extracted information and the technology process design rules, it transforms the layout into a constraint-based resizable symbolic template representation. The *"symbolic layout template"* is virtually an abstract representation of

the extracted layout properties, namely devices and connectivity, technology process design rules, and analog layout integrities.

The key tasks of the layout generator are to enforce new device sizes and to resolve the symbolic layout representation for rectangle locations. This is accomplished by a combination of linear programming and graph-based methods. The target layout is generated in CIF.

The direct incorporation of the embedded knowledge in the original layout as a template ensures retention of the layout integrity. The target layout generated by IPRAIL is checked for design-rule compliance.

4. LAYOUT TEMPLATE EXTRACTOR

The detailed flow of template extraction is shown in Figure 3. It involves representation of the layout in corner-stitching data structure, extraction of transistors, nets and passive devices, generation of layout constraints, and detection of device symmetry. Each sub-task performed by the template extractor is described below.

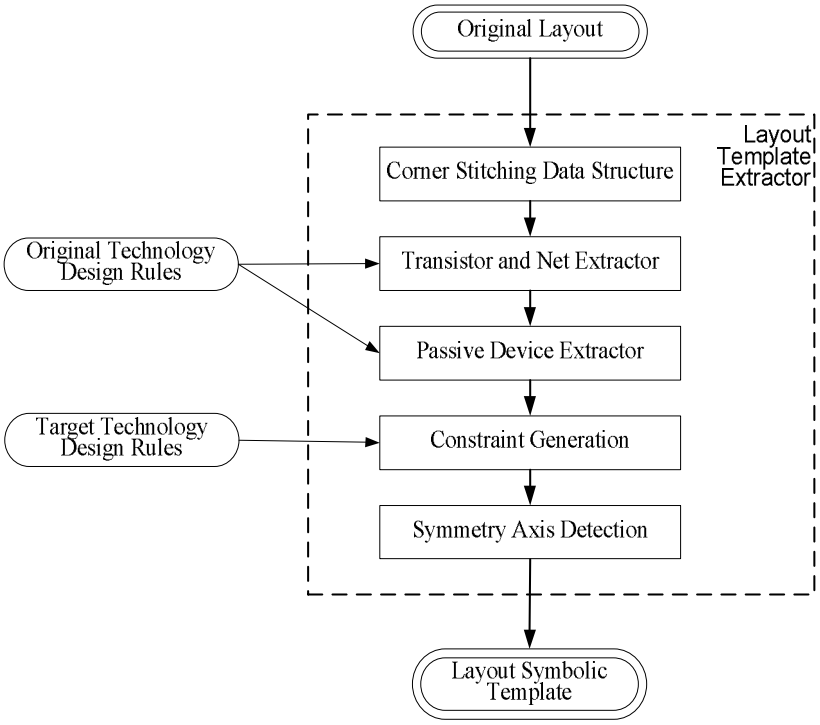


Figure 3: Internal flow of the layout template extractor in IPRAIL.

4.1 Layout Representation by Corner-Stitching Data Structure

IPRAIL adopts the *corner-stitching* data structure [15] for storing a layout. Our preference for corner-stitching over other potential data structures, for example bins and

linked-lists, is dictated by its efficiency in fast localized searches, as described in [16]. An example of the corner-stitching data structure is shown in Figure 4. In this scheme, the entire plane of each mask layer is represented explicitly in terms of solid and space rectangles called *tiles*. Each tile in a layer plane is connected to other tiles in the same plane by four *stitches* on its lower-left and upper-right corners. Some of the basic corner-stitching based operations frequently used in IPRAIL are *area-enumeration* for finding all tiles in a given area, *point-finding* for locating a tile at a given position, and *neighbor-finding* for listing all tiles adjacent to a given tile.

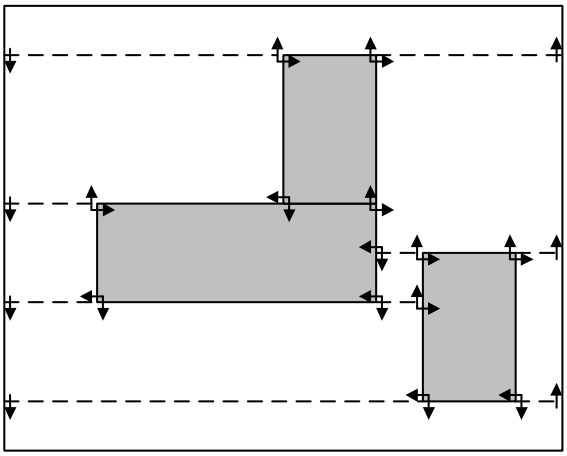


Figure 4: An example of the representation of a layout (one mask layer) in the corner-stitching data structure. Both solid tiles (gray) and space tiles (white) are organized as 'maximal horizontal strips'. All tiles are linked together by pointers at their corners.

4.2 Transistor and Net Extraction

A MOSFET transistor in a layout is defined as an overlap between two tiles in *poly-silicon* and *diffusion (active*) mask layers. A transistor has three terminals, viz., the gate terminal in the poly-silicon layer and the source and drain terminals in the diffusion layer. A net is defined as an electrical connection between the terminals of transistors or external ports.

Extraction of transistors and nets from the layout follows the algorithm proposed in the *Magic* VLSI layout system [17]. The usage of the corner-stitching data structure ensures fast extraction of transistors and nets in the circuit layout. Using area enumeration in the corner-stitching database, the extractor detects transistors by looking for overlaps between the poly-silicon and the diffusion layers. The transistor description stored in the database includes its orientation, size, location, and terminal information.

Once the transistors are extracted, a simple recursive algorithm detects nets in the layout using the terminals of the transistors as starting points. The fundamental operation involves marking all tiles that are electrically connected. The tile at the start-point is marked first and all neighbor tiles in the same mask layer are identified. The algorithm proceeds to one of the neighboring tiles and continues through a depth-first-search. If vias or contacts are encountered, the search moves to the next mask layer.

4.3 Extraction of Passive Devices

Resistors in analog layouts are typically designed in the poly-silicon mask layer as it exhibits high linearity, low capacitance to substrate and relatively small mismatches [3]. In some technologies, resistors are also constructed in the n-well or diffusion layers. The resistor topology supported in IPRAIL consists of single unit or multiple units laid out in parallel and connected in series, as shown in Figure 5.

Resistors in the layout are detected by searching through the tiles of the nets in the circuit. A single tile or a series of connected tiles of a net are classified as a resistor when the resistive value exceeds a user-defined threshold. Once a resistor is detected, its parent net is split into two. Connectivity between a resistor and the nets is maintained through port tiles. The resistor data structure stores geometry information along with connectivity at its ports.

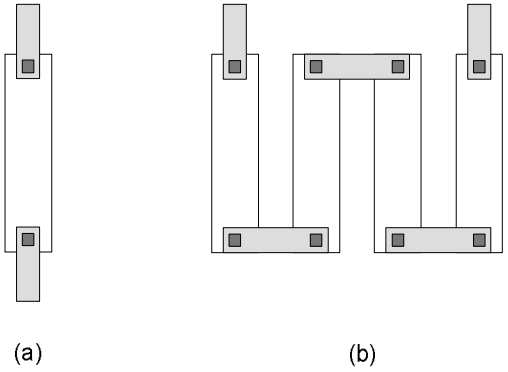


Figure 5: Layouts of resistors. White rectangles represent poly-silicon layer. (a) A resistor constructed from a single tile. (b) A resistor constructed from multiple series-connected tiles.

High-density linear capacitors are fabricated using a poly-silicon over another poly-silicon (*P-P*) layer in a "double poly" process [3]. In absence of such structures, capacitors are fabricated as sandwiches of two or more metal layers (*M-I-M*). Examples of P-P and M-I-M capacitors are shown in Figure 6. Alternately, MOS transistors can be used as non-linear capacitors (MOSCAP) by shorting their source and drain terminals.

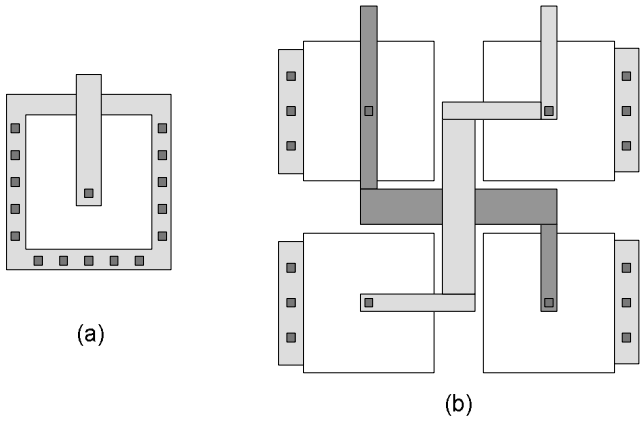


Figure 6: Layouts of P-P or M-I-M capacitors. (a) A capacitor constructed with one pair of tiles. (b) Two capacitors laid out in common-centroid configuration.

In IPRAIL, capacitors are defined as overlaps of two tiles in different layers that belong to different nets. Searching through the nets, the extractor detects capacitors when the capacitance due to the overlap exceeds a user-defined threshold. The MOSCAPs are detected during the transistor extraction. The geometry information and connectivity are stored in the capacitor data structure.

During the ensuing layout generation phase, if a passive is resized, other devices or wiring tiles may overlap with it. To prevent this, a *temporary dummy tile* is placed over the device location. The tile is defined in a new *dummy mask layer*, and is furnished with spacing constraints to every mask layer. This reserves exclusive space for the passive device.

4.4 Constraint Generation for Technology Migration

The symbolic template is based on a set of *geometric constraints* between the tiles in the layout. The constraints arise due to the connectivity between tiles and the technology design rules. The connectivity-based constraint between a pair of tiles ensures that the tiles remain electrically connected after the layout generation process. The constraints enforced by the technology design rules belong to one of the following three categories: (1) *minimum width* of a tile, (2) *minimum spacing* between two electrically unconnected tiles on the same or different mask layers, and (3) *minimum extension* of two overlapping tiles on different mask layers.

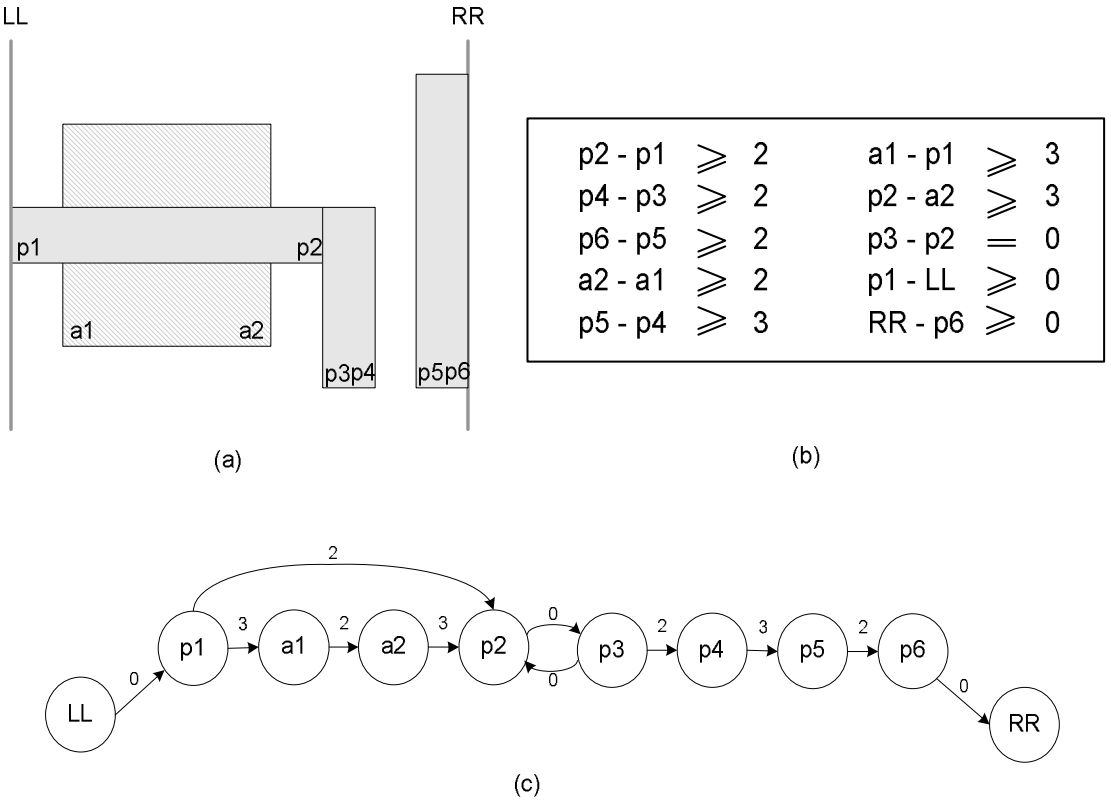


Figure 7: Constraint-based template formulation in horizontal direction. (a) An example layout. LL represents the left-most boundary and RR represents the right-most boundary. (b) A layout template in an equation form. (c) A layout template in a constraint graph form. The constraint values are taken from TSMC 0.25um CMOS technology process and are in scale of 0.15um.

The constraints for the symbolic template are established independently in the horizontal and vertical directions. Here, we describe the method for generating the constraints in the horizontal direction. First, we associate variables to the left and right *tile-edges* of each rectangle. The template constraints can be formulated in equation form. Figure 7(a) and Figure 7(b) illustrates a layout and its corresponding *constraint-equations*. From the example, in TSMC 0.25um CMOS technology process, the four different types of constraint-equations are:

$p_2 - p_1 \geq min\_poly\_width$	minimum width constraint
$p_5 - p_4 \geq min\_poly\_space$	minimum spacing constraint
$p_2 - a_2 \geq min\_diff\_extension$	minimum extension constraint
$p_3 - p_2 = 0$	connectivity constraint

Generating constraints between every pair of tile-edges in the layout leads to significant redundancy. For example, in Figure 8, the minimum spacing constraint between the right tile-edge of rectangle *b (represented by* b_r*)* and the left tile-edge of rectangle *s (represented by* s_l*)* is not required. This is due to the existence of a minimum width

constraint in a tile d, and minimum spacing constraints between tile-edges b_r and d_l and between tile-edges d_r and s_l. This is verified from the following constraint equations:

$$d_l - b_r \geq min\_space \quad (1)$$
$$s_l - d_r \geq min\_space \quad (2)$$
$$d_r - d_l \geq min\_width \quad (3)$$
$$(1)+(2)+(3): \quad (d_l - b_r) + (s_l - d_r) + (d_r - d_l) \geq 2(min\_space) + min\_width$$
$$\text{or:} \quad s_l - b_r \geq 2(min\_space) + min\_width \quad (4)$$

Equation (4) is the linearly dependent on the equations (1-3) and therefore superfluous. To prevent such redundancy, the constraint equations are generated by employing a *scan-line* method [19].

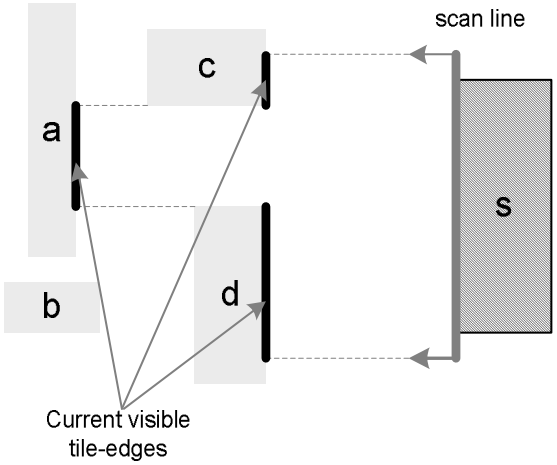


Figure 8: Example of the scan-line method. The current visible tile-edges consist of part of the right tile-edge of *a*, all of the right tile-edge of *c* and all of the right tile-edge of *d* (as marked). After the tile *s* is processed, the visible tile-edges will consist of all the right tile-edge of *s* and parts of the right tile-edges of *c* and *d* that are not blocked by the tile *s*.

The scan-line used for generating horizontal constraints is a vertical line that sweeps through the layout from left to right. It enumerates all tile-edges in a list sorted by their abscissas. For a tile-edge in the sorted list, constraints are generated from that tile-edge to all other tile-edges to its left that are "visible" (not blocked by other tiles in the same mask layer) from it. For example in Figure 8, parts of the right tile-edges of the rectangles *a*, *c* and *d* are visible from the left tile-edge of rectangle *s* while the right tile-edge of rectangle *b* is not visible as it is blocked by rectangle *d*. Thus, while processing the left tile-edge s_l of rectangle *s*, separation constraints are generated only for the right tile-edges a_r, c_r and d_r of rectangles *a*, *c* and *d* respectively. The process continues until all edges are handled. The algorithm has a worst-case complexity $O(N^2)$, where N is the number of tiles.

The constraint-equations of the symbolic template, as defined in Figure 7(a) and Figure 7(b), can also be represented as a *weighted directed constraint graph* $G = [V, E]$. From the constraint-equations, a graph can be constructed where the equation variables

are represented as graph vertices *(V)* and the equation constants as weights of the graph edges *(E)*. All connectivity and design rule constraints appear in one of the three following forms: *lower bound constraints, upper bound constraints* and *fixed weight constraints*. A lower bound constraint-equation of the form $x_i - x_j \geq w$ is represented in the graph as a directed edge from vertex x_j to vertex x_i of weight w. An upper bound constrained equation of the form $x_i - x_j \leq w$ is represented by a directed edge from vertex x_i to vertex x_j of weight $-w$. A fixed weight constrained equation of the form $x_i - x_j = w$ is first separated into two different equations $x_i - x_j \geq w$ and $x_j - x_i \geq -w$. These are represented in the graph by one directed edge of weight w from vertex x_j to vertex x_i and another directed edge of weight $-w$ from vertex x_i to vertex x_j respectively. Figure 7(c) shows an example of the graph constructed from the equations.

Here, if the target layout design involves a migration to a new process technology, the constraint-equations have to be updated with the minimum width, spacing and extension design rules of the new technology.

4.5 Symmetry Detection

Detection of matching and symmetry of transistors in the layout is of utmost importance for layout template extraction. Two transistors are considered symmetric if they are "*geometrically mirrored*". This involves equal dimension, similar horizontal or vertical location, uniform orientation and same type.

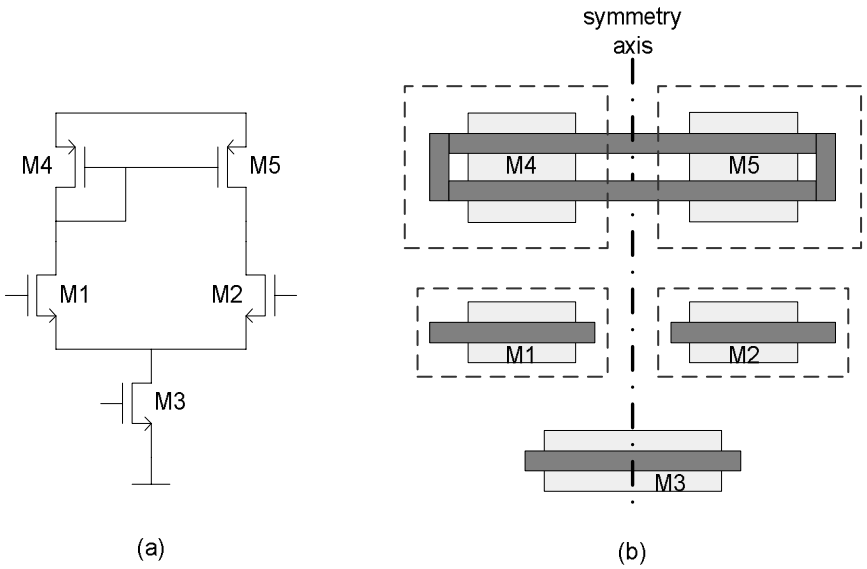


Figure 9: Example of symmetric transistors. (a) The symmetric pairs are (M1:M2) and (M4:M5). (b) The symmetry axis of transistor pairs M1:M2 and M4:M5. The layout in (b) is a simplified version and only CMOS diffusion and poly-silicon layers are shown.

The symmetry detection in IPRAIL is based on the algorithm described in [20]. Here, we describe the detection for horizontally aligned transistors. Consider the two

transistors *M4* and *M5* in Figure 9. First, all tiles belonging to transistor *M4* are collected in a list in an increasing order of the left tile-edge positions. The tiles that constitute transistor *M5* are collected in another list in a decreasing order of the right tile-edge positions. If the contents of the two lists are matched in terms of locations, sizes, and layers, then the two transistors are deemed symmetric and a symmetry axis is detected. The symmetry axis for vertically aligned transistors can be obtained similarly. As for the common-centroid topology in Figure 10, both horizontal and vertical symmetry axes are detected.

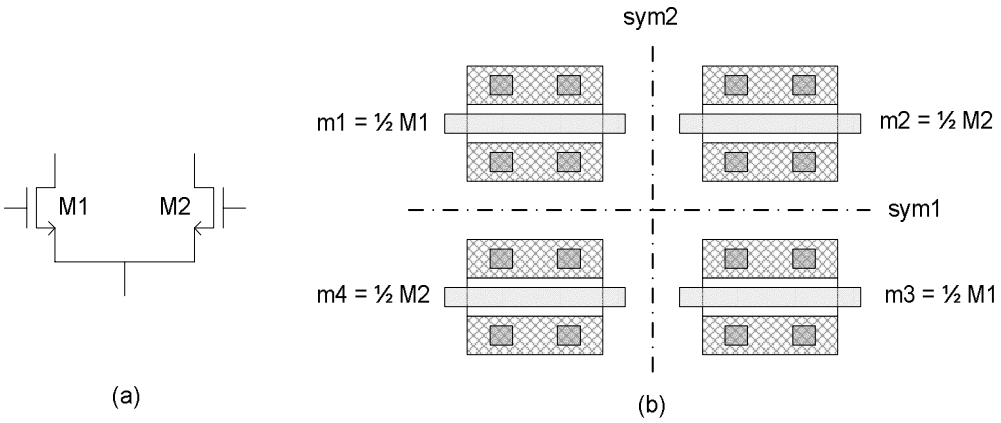


Figure 10: A common-centroid layout and symmetry axes. (a) Schematic diagram. Here transistors M_1 and M_2 are matched. (b) Layout. The transistors M_1 and M_2 are each divided into two halves. Each half is laid out diagonally from the other to cancel out mismatches. Here, two symmetry axes are required to maintain common-centroid layout. Transistor pairs (m_1:m_4) and (m_2:m_3) are symmetric by a sym_1 axis. Transistor pairs (m1:m2) and (m4:m3) are symmetric by a sym_2 axis.

We extend our algorithm to the detection of symmetry between two groups of transistors. Two groups are symmetric if there exists inter-group pair-wise matching between horizontally or vertically aligned transistors. In some circuits, designer intervention might be necessary for detecting matching between only a few groups of transistors. In this interactive mode, symmetry between two groups of transistors can be detected by drawing bounding-boxes thereby selecting each group. In IPRAIL, similar facility for detecting symmetry in the batch mode is also provided through input text files.

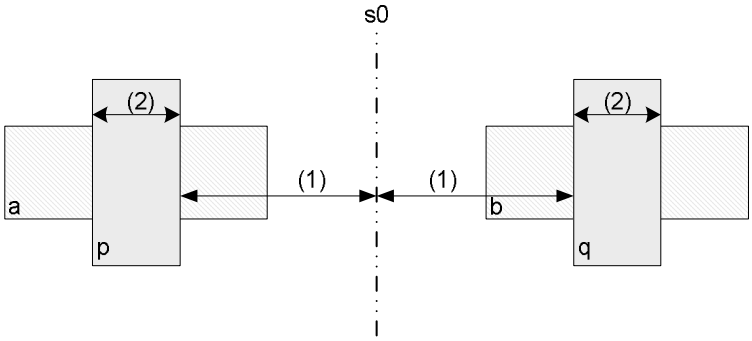


Figure 11: A simplified layout of two transistors showing only diffusion (stripes) and poly-silicon (gray) layers. Edges of rectangle *'a'* are defined as *'a_l'* (left), *'a_r'* (right), *'a_b'* (bottom) and *'a_t'* (top). Edges of rectangles *'b'*, *'p'* and *'q'* are defined similarly. The symmetrical axis is denoted by *'s_0'*. The constraints generated due to symmetrical transistors are

(5) $(s_0 - p_r) - (q_l - s_0) = 0$ **enforces equal distance from transistors to the symmetry axis.**
(6) $(p_r - p_l) - (q_r - q_l) = 0$ **enforces equal length of transistors.**
(7) $a_t - b_t = 0$ **and** $a_b - b_b = 0$ **enforce the vertical location and the width of transistors.**

For each symmetric transistor pair, three types of constraint-equations are generated as illustrated in Figure 11. Equation (5) *"$2s_0 - p_r - q_l = 0$"* preserves the distance between left and right transistors to the symmetry axis. Equation (6) *"$p_r + q_l - p_l - q_r = 0$"* matches the lengths of both transistors. As these three-variable or four-variable constrained equations maintain the distance between two variable pairs, we shall call them *equi-distance constraints*. Equations (7) *"$a_t - b_t = 0$"* and *"$a_b - b_b = 0$"* maintain the vertical location of both transistors and match their widths.

As for representation in the constraint graph, the fixed-weight equations *"$a_t - b_t = 0$"* and *"$a_b - b_b = 0$"* can be included in the graph directly (as described in Section 4.4). However, there are no straightforward ways to represent the equi-distance equations as graph edges. So these constraints have to be set aside in the equation form.

5. LAYOUT GENERATOR

The layout generator in IPRAIL creates the target layout from the symbolic template extracted from the original layout. Figure 12 shows the various steps in the layout generation process. First, the layout generator updates the template with the transistor and passive sizes obtained from a circuit synthesis tool. As the updated template consists of the symmetry, connectivity and design rule constraints, the problem of generation of the target layout from the symbolic template reduces to a *symbolic compaction problem*. This is solved by a combination of linear programming and graph-based shortest-path algorithm.

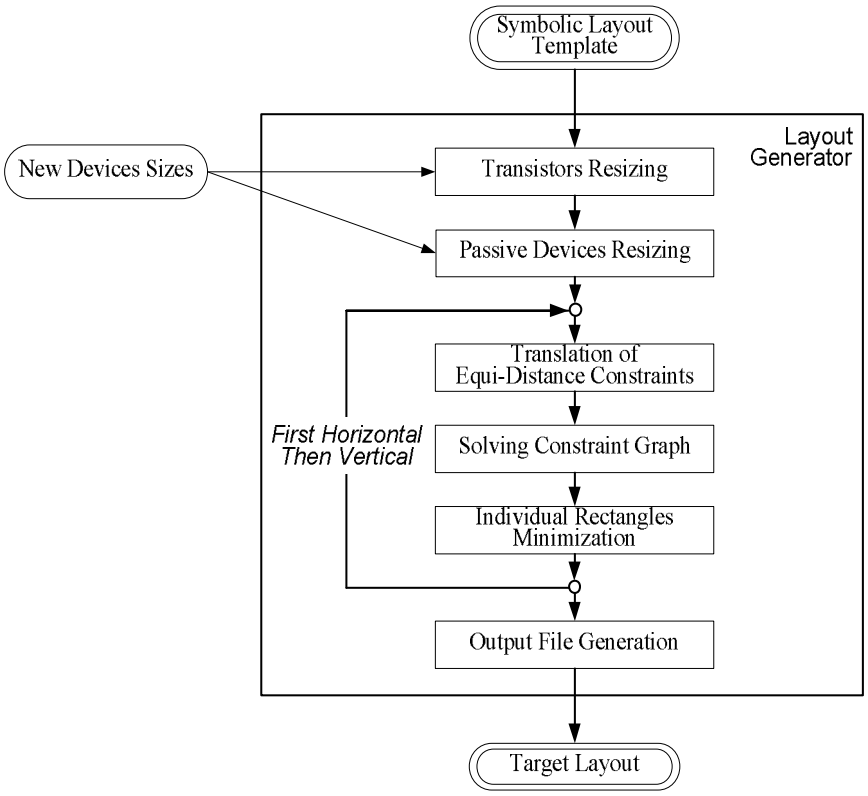


Figure 12: Internal flow of the layout generator in IPRAIL.

5.1 Transistor Sizing

For a new technology process, the new transistor widths and lengths can be added to the constraint graph as fixed weight constraints on the gate diffusion tile and gate poly-silicon tile, respectively. Here, due to changes in transistor sizes, the diffusion-metal *contacts* at the drain and source terminals require careful handling. If the size of a transistor is smaller in the target layout than the original, the drain or source area may not be able to accommodate the original number of contacts. To overcome this, the contacts are first removed from the layout, and extra constraints are added between the diffusion and metal tile-edges to preserve their overlap. Figure 13 illustrates the constraints of the transistors in horizontal orientation. Figure 14 illustrates the constraints of the transistors in vertical orientation. During the generation of the target layout upon the completion of the problem, rows of contacts are added to connect such diffusion and metal tiles.

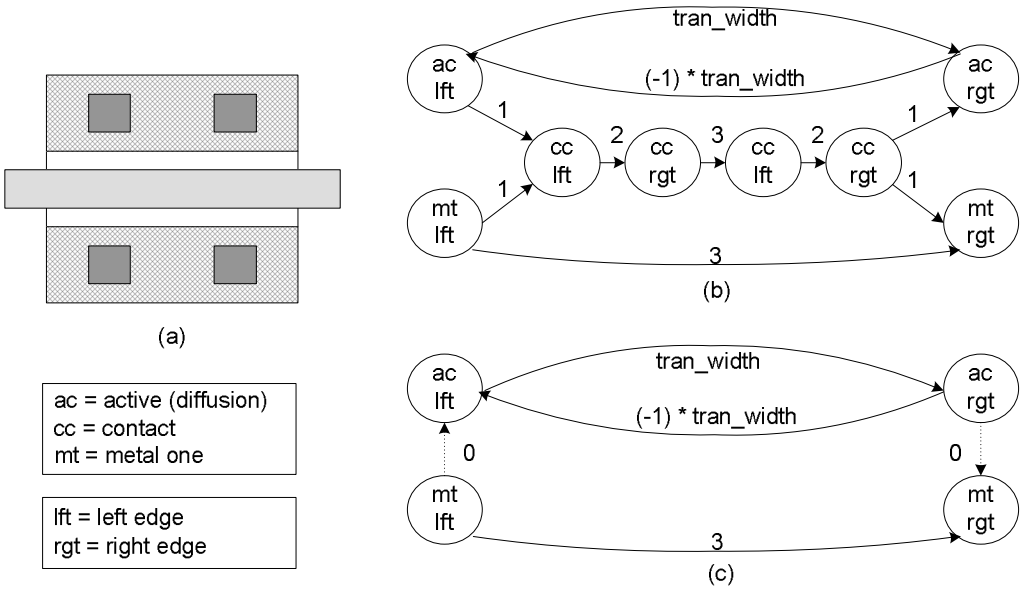


Figure 13: Contact removal in transistor resizing along the width of transistors. (a) A transistor layout. (b) Original constraint graphs for only lower diffusion side with contacts. (c) After removing contacts, two constraints are added for connectivity (metal->diffusion and diffusion->metal). Note: Constraints weights are taken from TSMC 0.25um process and in unit of process lambda (0.15um).

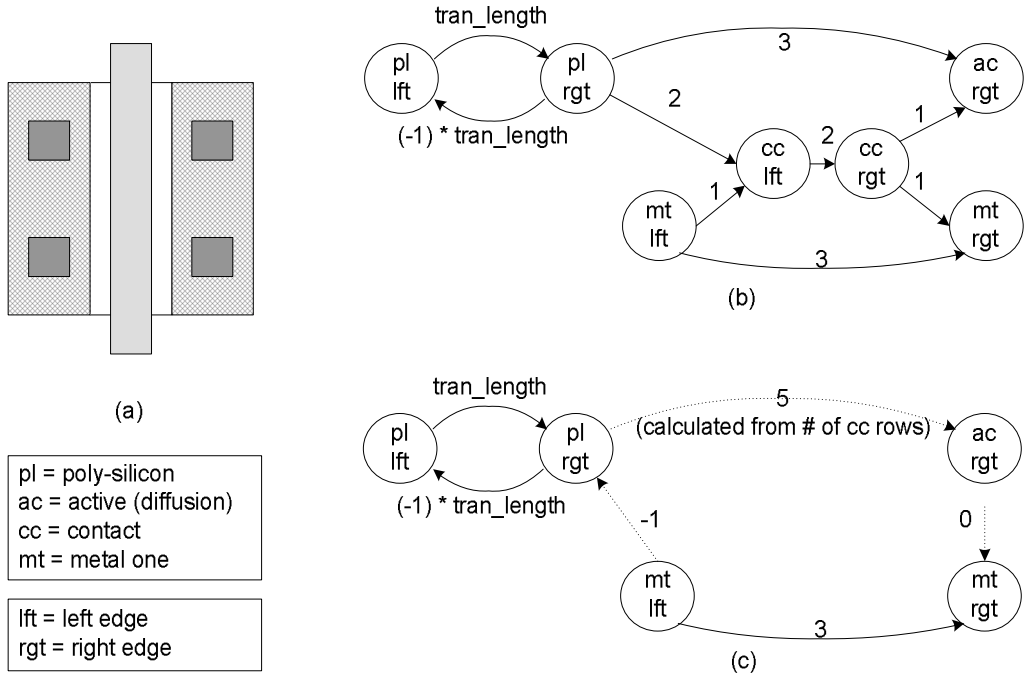


Figure 14: Contact removal in transistor resizing along the length of transistors. (a) A transistor layout. (b) Original constraint graphs for only right diffusion side with contacts. (c) After removing contacts, two constraints are added and one constraint is updated for connectivity. Note: Constraints weights are taken from TSMC 0.25um process and in unit of process lambda (0.15um).

5.2 Updating Passive Device Sizes

For a new technology process, the passive device geometries and dimensions can be changed from its original layout. Updating the passive device sizes is done by adding constraints between the temporary dummy tile left and right tile-edge variables. Furthermore, extra constraints have to be incorporated to the constraint graphs between the temporary dummy tile variables and the port tile variables to keep the passive device aligned. This maintains connection between the passive devices and the circuit nets after the compaction process. An example of temporary dummy tile variables, port tile variables and their constraints is described in Figure 15.

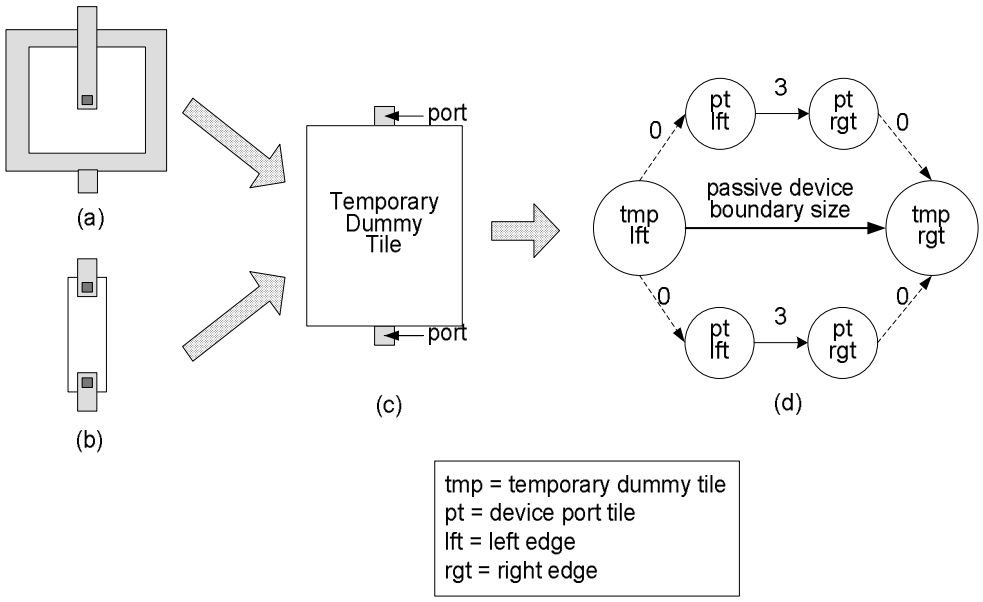


Figure 15: Passive devices resizing. (a) A P-P or M-I-M capacitor layout. (b) A resistor layout. (c) Replacing the passive device with a temporary dummy tile. (d) Constraint graphs. The minimum width of port layer in this example is arbitrarily chosen as 3 units.

There are two schemes for replacing a passive device. If the new device has the same structure as the original one but is different in size, the fixed-weight constraints are added between corresponding tile variables in the graph. If the new device has different structure, we can totally remove the passive device tile variables and their constraints. The dummy tile will be resized according to the new boundary geometries. In this case, when generating the output layout, the new device layout has to be created and added in place of the dummy tile.

5.3 Compaction by Combined Linear Programming and Graph-Based Algorithms

The assembled template constraint problem can be solved by applying the layout compaction algorithm. The problem in equation form can be solved directly using *linear programming* (LP) [21]. Nevertheless, it is too computationally expensive for VLSI

layouts, due to the problem size. Traditionally, graph-based compaction methods [19] are preferred for their superior computational speed. The presence of the equi-distance constraints resulting from symmetry detection, however, hinders the successful application of the conventional graph-based methods. Recall the example layout of Figure 11. The two equi-distance equations for the layout are

$$2s_0 - p_r - q_l = 0 \quad (5)$$
$$p_r + q_l - p_l - q_r = 0 \quad (6)$$

Equation (5) preserves the distance from the left and the right transistors to the symmetry axis. The matching of the two transistor lengths is ensured by Equation (6). As the equi-distance equations contain three or four variables, they cannot be directly incorporated as weighted edges into the constraint graph. Therefore, all equi-distance constraints have to be converted into combinations of two-variable constrained equations, which can be added to the graph. For this, we use a combination of linear programming and graph-based algorithms introduced in [24].

The original constraint-graph $G = [V, E]$ is first reduced to a smaller *core-graph* $G' = [V', E']$, where $V' = \{v_{LL}, v_{RR}\} \cup \{ v_i | x_i$ is the corresponding variable that appears in the equi-distance constraints $\}$. The vertices v_{LL} and v_{RR} correspond to the left and the right boundaries respectively. For $\forall v_i, \forall v_j \in V' | i \neq j$, we search for the longest directed path from v_i to v_j in graph G. If the longest path exists, a directed edge $e = \langle v_i, v_j \rangle$ is then added to graph G' and the weight of the edge is set to the length of the path from v_i to v_j in G. At this point, the core-graph G' possesses the same constraint properties as the initial graph G. LP-compatible equations generated from the reduced constraint graph G' and the equi-distance constraints are then solved together by linear programming.

The solution from the LP problem above yields optimal values for the variables related to the equi-distance constraints. This, in effect, converts the equations into a form that can be added to the original constraint graph G. We obtain the following form for equation (5), where b is a constant calculated from the optimal linear programming solution.

$$s_0 - p_6 = p_7 - s_0 = b \quad (8)$$

Equation (8) can then be further converted into the following form.

$$s_0 - p_6 \geq b \ , \ p_6 - s_0 \geq -b \ , \ p_7 - s_0 \geq b \ , \ s_0 - p_7 \geq -b \quad (9)$$

Directed edges of weight b are then added from p_6 to s_0 and from s_0 to p_7 in the original constraint graph G. Similarly, directed edges of weight $-b$ are added from

s_0 to p_6 and from p_7 to s_0 in G. In this way, a complete constraint graph incorporating the equi-distance constraints is constructed. And finally, we solve the compaction problem by applying the shortest path algorithm on the complete constraint graph. In IPRAIL, the *Bellman-Ford algorithm* [25] is chosen due to its ability to handle negative-weight edges. The Bellman-Ford algorithm has the worst-case complexity of *O(VxE)*, where *V* is the number of vertices and *E* is the number of edges [26].

5.4 Minimization of Individual Rectangles

The solution obtained from the shortest path algorithm can cause another problem. The shortest path algorithm finds the minimum distance from every variable to the left-most variable, which is the left boundary. This causes some tiles to extend excessively toward its left. An example is illustrated in Figure 16. Such extensions introduce unnecessary parasitic resistance and capacitance, which affect the performance of the design adversely. Therefore, after the first feasible solution is obtained, the *individual rectangle minimization* algorithm needs to be performed.

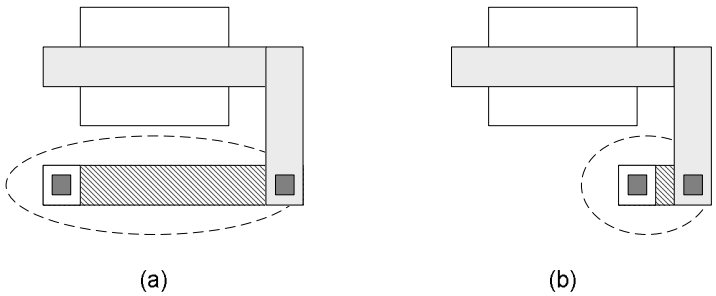


Figure 16: Example of tile extensions caused by the shortest path algorithm. Shown in circles are two metal-one rectangles: (a) obtained from the shortest path algorithm with left boundary as a source and is not minimized, and (b) attained after individual rectangle minimization is performed.

The individual rectangle minimization, applied in IPRAIL, is a modification of the *wire-length minimization* presented in [27]. First, the algorithm locates a *critical path*, which is the shortest path from the left to the right boundary in the constraint graph. The critical path can be determined by employing the shortest path algorithm twice; first from left to right and then from right to left. All variables that possess similar positions belong to the critical path. Tile variables in the path are already minimized and cannot be moved. The rest are indicated as movable tiles, whose areas will be reduced. The algorithm attempts to shift the movable tile variables to their feasible right-most locations, if the moves result in smaller overall tile area. To speed up the process, tile variables are dynamically grouped and ungrouped as they are repositioned towards the right. When the mobility of a tile variable is restricted by another tile variable in the critical path, its optimum position is reached and the variable is not considered for any further operation.

During the process, if moving a tile variable or a group of tile variables decreases the area of one tile and increases that of another, priority is given to the tile in the layer with larger resistance and capacitance. This is accomplished by assigning weight

coefficients to all variables in the graph. The algorithm ends if all tile variables are restricted by critical path or if moving the variables does not bring about any further reduction in total tile area.

Similar to the wire-length minimization algorithm, IPRAIL's individual rectangle minimization has the worst-case complexity of $O(N^2 \log N)$, where N is the number of constraint edges. The average-case complexity is almost $O(N)$.

5.5 Output CIF File Generation

The shortest path algorithm and the individual rectangle minimization are completed in both horizontal and vertical directions. They provide all rectangles their new positions in the target layout. As the resizing of transistors involves the removal of all diffusion-metal contacts, they are inserted back into the target layout. The number of contacts to be replaced is calculated based on the diffusion-metal overlap area and the design rules for contacts in the new technology process.

As for the passive devices, in case there is a change in device configuration, the original set of tiles is removed from the constraint problem. Thus, new tiles need to be re-constructed based on the new device geometries and device configurations. Finally, the tiles are inserted in their exact positions depending on the temporary dummy tiles. It may be necessary to rotate or flip the tile structures so the devices are aligned with their ports.

6. EXAMPLES OF LAYOUT RETARGETING USING IPRAIL

The IPRAIL-based methodology of retargeting layouts is presented for a single-ended folded-cascode operational amplifier and a two-stage Miller-compensated operational amplifier. Both circuit layouts are initially designed in the TSMC 0.25um CMOS process and are retargeted to the TSMC 0.18um CMOS process. The new device sizes are obtained from design and simulation of the circuit netlist in the target process.

For both examples, the layout description files in CIF and the original and target technology design rules from Cadence environment are imported to IPRAIL. Once the retargeting process is finished, the regenerated layouts are design-rule checked (DRC). Both original and target layout netlists are extracted and simulated in Hspice to compare their functionalities and performances.

6.1 Single Ended Folded Cascode Operational Amplifier

Figure 17 illustrates the schematic of a single-ended folded-cascode operational amplifier. The design consists of 14 transistors. Figure 18 shows the original layout in TSMC 0.25um CMOS process. The transistors are represented in multi-finger structures, which cause the layout to contain 43 distinct transistors.

The target layout is generated by IPRAIL, focusing on three main factors. First, three symmetrical axes between transistors are taken into account, depicted as *A*, *B* and *C* in the layout. Second is a set of design rules in TSMC 0.18um CMOS process. Last, the new transistor sizes are compiled based on the evaluation and simulation of the schematic netlist in the new process, such that the desired specifications are met.

We employ IPRAIL to retarget the original layout following two different schemes; first, keeping the original device sizes (denoted as original-device-size), and then, imposing the new device sizes (denoted as new-device-size), listed in Table 1. In both schemes, the symmetry axes and the new technology process are supported. The result of original-device-size layout and new-device-size layout are presented in Figure 19 and Figure 20 respectively. The statistics on the performances and silicon areas are summarized in Table 2.

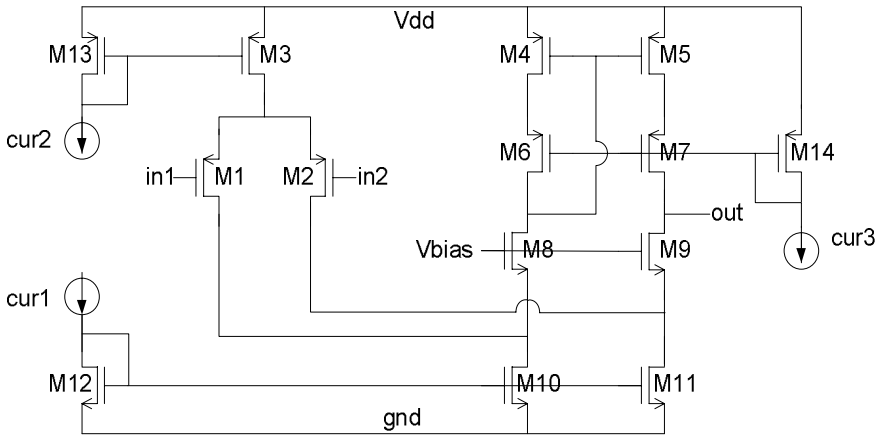


Figure 17: Schematic of a single-output folded-cascode opamp.

Table 1: Transistors sizes of a folded-cascode opamp.

Transistors	*total width (um) / total length (um)*		
	0.25um	*0.18um original-device-size*	*0.18um new-device-size*
M1, M2	*48.0 / 1.2*	*48.0 / 1.2*	*33.6 / 1.4*
M3, M13	*96.0 / 1.2*	*96.0 / 1.2*	*56.0 / 1.4*
M4, M5	*63.6 / 1.2*	*63.6 / 1.2*	*15.2 / 0.9*
M6, M7	*63.6 / 1.2*	*63.6 / 1.2*	*15.2 / 0.9*
M8, M9	*31.2 / 1.2*	*31.2 / 1.2*	*6.6 / 1.3*
M10, M11	*41.4 / 1.2*	*41.4 / 1.2*	*36.4 / 1.3*
M12	*41.4 / 1.2*	*41.4 / 1.2*	*36.4 / 1.3*
M14	*13.8 / 1.2*	*13.8 / 1.2*	*6.0 / 0.9*

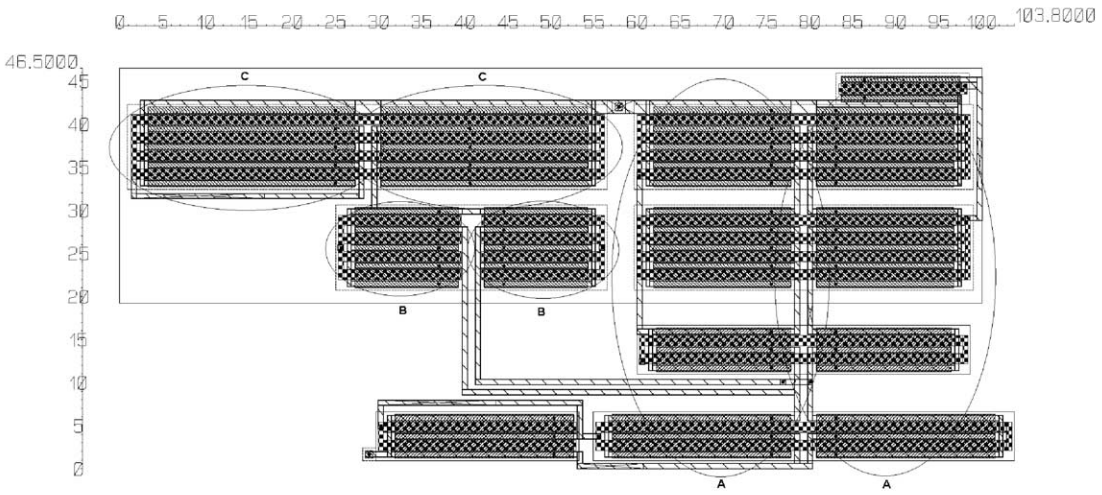


Figure 18: Original layout of a folded cascode opamp in TSMC 0.25um. A, B and C are symmetrical transistor block pairs.

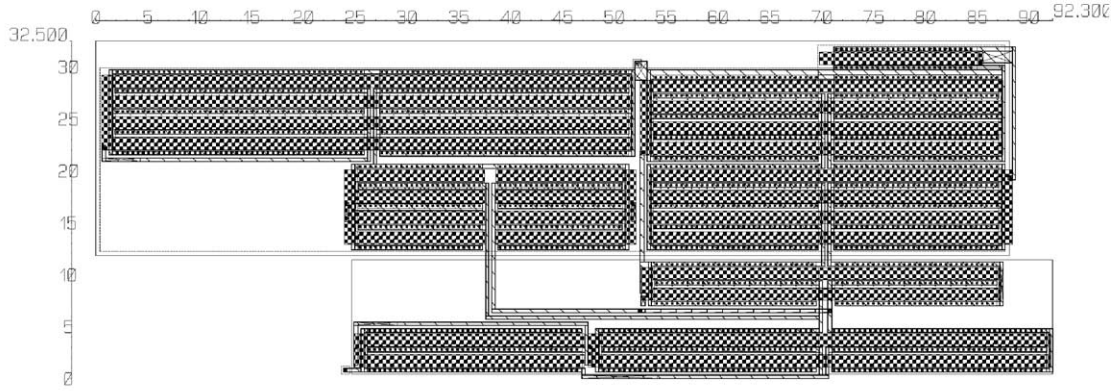


Figure 19: Target layout of a folded cascode opamp in TSMC 0.18um. Original transistor sizes are retained.

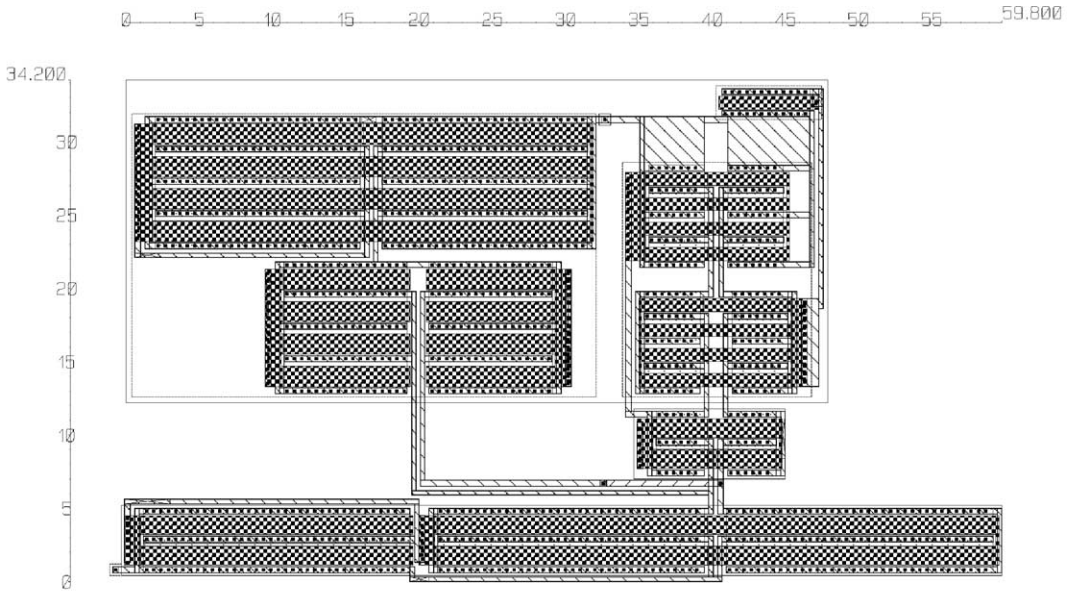


Figure 20: Target layout of a folded cascode opamp in TSMC 0.18um. Transistors are resized according to Table 1.

Table 2: Performances comparison of a folded-cascode opamp.

	0.25um	*0.18um original-device-size*	*0.18um new-device-size*
Vdd	*2.5 V*	*1.8 V*	*1.8 V*
Load Cap.	*1.0 pF*	*0.7 pF*	*0.7 pF*
Gain	*60.9 dB*	*61.9 dB*	*60.6 dB*
Bandwidth	*51.7 MHz*	*71.7 MHz*	*63.5 MHz*
Phase Margin	*63 deg*	*42 deg*	*71 deg*
Gain Margin	*12.5 dB*	*12.4 dB*	*10.5 dB*
Power	*1.48 mW*	*1.07 mW*	*0.88mW*
Area	*4826.70 um^2*	*2995.75 um^2*	*2045.16 um^2*

6.2 Two-Stage Miller-Compensated Operational Amplifier

Figure 21 shows the two-stage Miller-compensated operational amplifier that consists of 8 transistors. Its original layout in TSMC 0.25um CMOS process is illustrated in Figure 22. The compensation capacitor is designed by using the MOSCAP. With the multi-finger structures in both transistors and MOSCAP, the layout contains 48 distinct transistors.

Similar to the folded-cascode opamp, we employ IPRAIL twice; first with original-device-size, and then, with new-device-size, whose dimensions are listed in

Table 3. The target layouts in TSMC 0.18um CMOS process are illustrated in Figure 23 and Figure 24 for the original-device-size and new-device-size respectively. Table 4 summarizes the performances and area comparison.

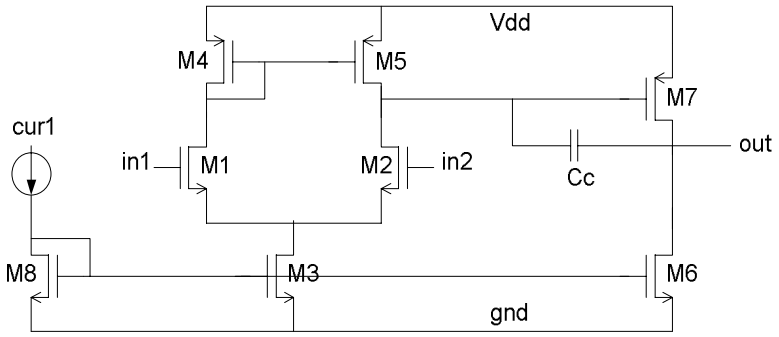


Figure 21: Schematic of a two-stage Miller-compensated opamp.

Table 3: Transistors sizes of a two-stage opamp.

Transistors	*total width (um) / total length (um)*		
	0.25um	*0.18um original-device-size*	*0.18um new-device-size*
M1, M2	*90.0 / 0.3*	*90.0 / 0.3*	*80.0 / 0.3*
M3	*21.5 / 0.6*	*21.5 / 0.6*	*21.0 / 0.6*
M4, M5	*17.1 / 0.3*	*17.1 / 0.3*	*17.1 / 0.3*
M6	*90.0 / 0.6*	*90.0 / 0.6*	*90.0 / 0.6*
M7	*420.0 / 0.6*	*420.0 / 0.6*	*210.0 / 0.4*
M8	*6.0 / 0.6*	*6.0 / 0.6*	*6.0 / 0.6*

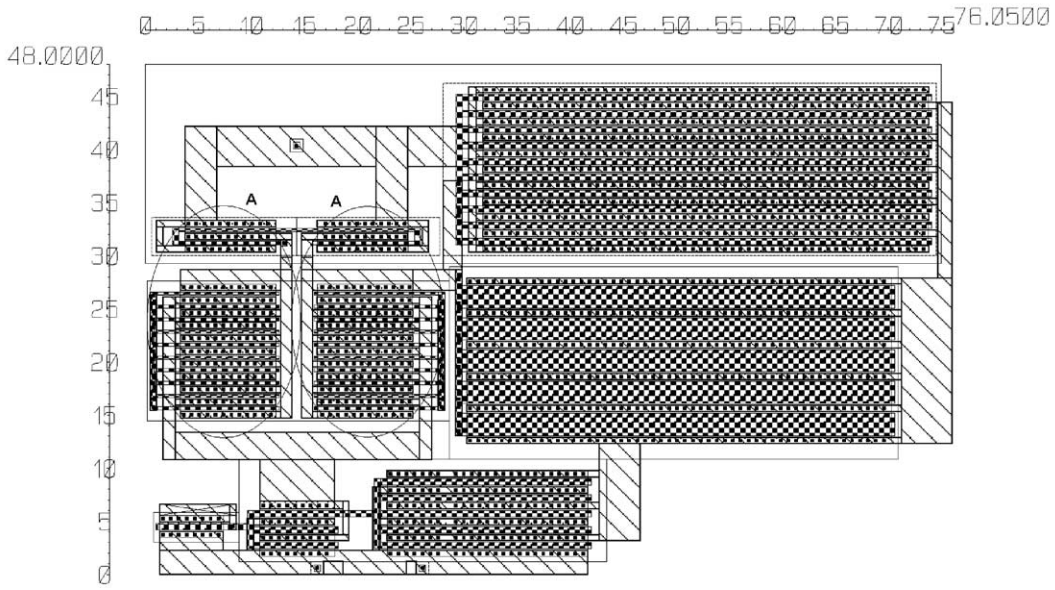


Figure 22: Original layout of a two-stage opamp in TSMC 0.25um. A is a symmetrical transistor block pairs.

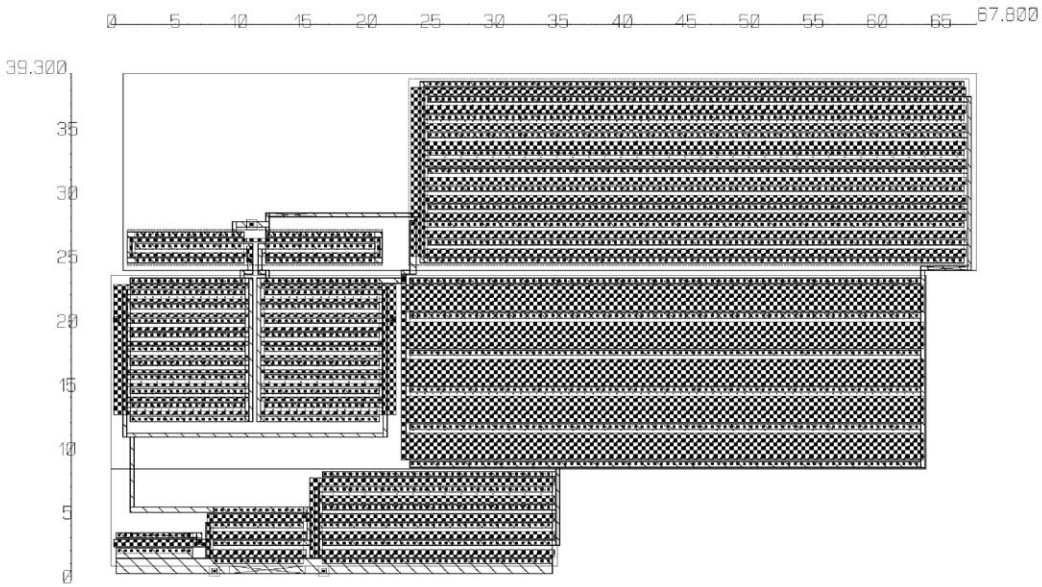


Figure 23: Target layout of a two-stage opamp in TSMC 0.18um. Original transistor sizes are retained.

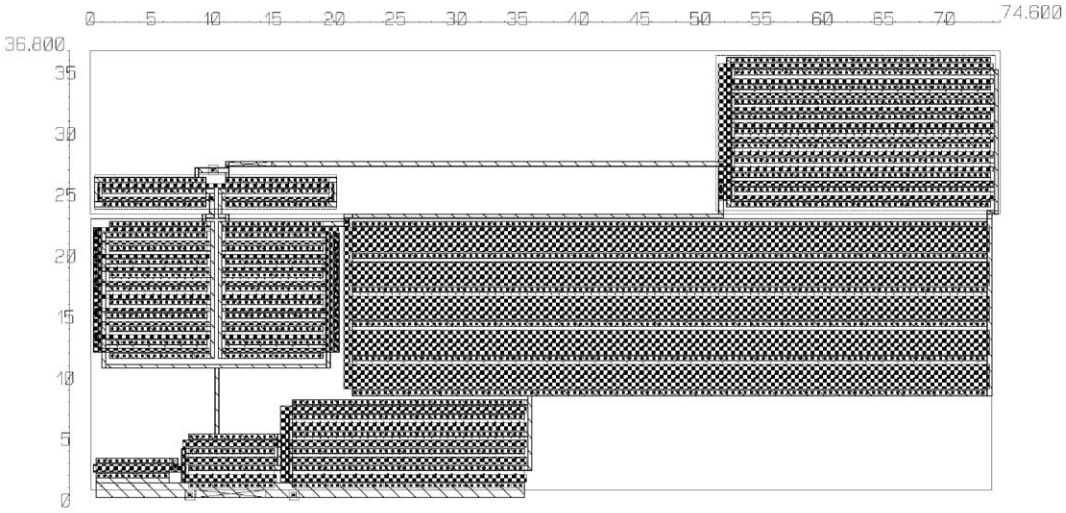


Figure 24: Target layout of a two-stage opamp in TSMC 0.18um. Transistors are resized according to Table 3.

Table 4: Performances comparison of a two-stage opamp.

	0.25um	*0.18um original-device-size*	*0.18um new-device-size*
Vdd	*2.5 V*	*1.8 V*	*1.8 V*
Load Cap.	*1.0pF*	*0.7pF*	*0.7pF*
Gain	*57.7 dB*	*39.6 dB*	*64.4 dB*
Bandwidth	*135 MHz*	*181 MHz*	*104 MHz*
Phase Margin	*50 deg*	*56 deg*	*56 deg*
Gain Margin	*9.6 dB*	*12.5 dB*	*9.2 dB*
Power	*4.82 mW*	*3.56 mW*	*3.46 mW*
Area	*3650.40 um^2*	*2664.54 um^2*	*2745.28 um^2*

The runtime on the folded cascade opamp is 39.2 seconds and on the two-stage opamp is 37.6 seconds on a 440MHz SUN Ultrasparc10 workstation. For each example, the time elapsed on both the original-device-size and new-device-size cases are the same.

7. LIMITATIONS OF IPRAIL

As the methodology in the current version of IPRAIL is based on recycling of the original layout, there are a few limitations. First, the target technology process has to cover all the layers presented in the original layout; this is what we called modestly new process migration. Second, the new device sizes given to the tool cannot be arbitrary,

and has to be resizable on the layout. In particular, if two transistors share the same drain diffusion rectangle, the widths of both transistors have to be the same. Otherwise, it will result in an over-constrained problem. Last, creating a symbolic template directly from the original layout may limit design configuration. If there is a change in voltage level in the target technology, it may adversely affect the performance of certain design topologies. In the current version of IPRAIL, we assume the circuit topology remains the same when migrating to a modestly new process. Despite these limitations, we have found that IPRAIL is a useful tool for a lot of practical retargeting problems.

8. CONCLUSIONS

An automatic analog layout tool, IPRAIL, which is capable of re-targeting the layout to different technology processes, is presented. Layout recycling through symmetry detection and layout integrity conservation scheme is used in order to preserve the analog layout property. Additionally, IPRAIL considers new device sizes to satisfy new specifications as part of the re-targeting process. IPRAIL has been applied successfully to migrate some practical analog circuit layouts.

ACKNOWLEDGEMENTS

The authors would like to thank Youcef Bourai and Bo Wan for their participation in the early phase of the IPRAIL project, and also Kiyong Choi and Jinho Park for valuable discussions on circuit examples.

REFERENCES

[1] N. Jangkrajarng, S. Bhattacharya, R. Hartono, and C-J. R. Shi, "Automatic Analog Layout Retargeting for New Processes and Device Sizes", *Proceedings of IEEE International Symposium on Circuits and Systems*, in press, May 2003.

[2] M. J. M. Pelgrom, A. C. J. Duinmaijer and A. P. G. Welbers, "Matching Properties of MOS Transistors", *IEEE Journal of Solid State Circuits*, vol. 24, pp. 1433-1440, October 1989.

[3] A. Hastings, *The Art of Analog Layout*, Prentice Hall Incorporate, 2001.

[4] B. Razavi, *Design of Analog CMOS Integrated Circuits*, McGraw Hill, 2001

[5] H. Koh, C. Sequin and P. Gray, "OPASYN: A Compiler for CMOS Operational Amplifiers", *IEEE Transactions on Computer-Aided-Design of Integrated Circuits and Systems*, vol. 9, pp. 113-125, February 1990.

[6] H. Onodera, H. kanbara and K. Tamaru, "Operational-Amplifier Compilation with Performance Optimization", *IEEE Journal of Solid State Circuits*, vol. 25, pp. 466-473, April 1990.

[7] J. D. Conway and G. G. Schrooten, "An Automatic Layout Generator for Analog Circuits", *Proceedings of European Design Automation Conference*, pp. 513-519, March 1992.

[8] R. Castro-Lopez, F. V. Fernandez, F. Medeiro and A. Rodriguez-Vazquez, "Generation of Technology-Independent Retargetable Analog Blocks", *International Journal of Analog Integrated Circuits and Signal Processing*, Kluwer Academic Publishers, vol. 33, pp. 157-170, December 2002.

[9] M. Aktuna, R. A. Rutenbar, and L. R. Carley, "Device-Level Early Floorplanning Algorithms for RF Circuits", *IEEE Transactions on Computer-Aided-Design of Integrated Circuits and Systems*, vol. 18, no. 4, pp. 375-388, April 1999.

[10] J. M. Cohn, D.J. Garrod, R. A. Rutenbar and L. R. Carley, "KOAN/ANAGRAM II: New Tools for Device-Level Analog Placement and Routing", *IEEE Journal of Solid State Circuits*, vol. 26, pp. 330-342, March 1991.

[11] K. Lampaert, G. Gielen and W. M. Sansen, "A Performance-Driven Placement Tool for Analog Integrated Circuits", *IEEE Journal of Solid State Circuits*, vol. 30, pp. 773-780, July 1995.

[12] E. Malavasi, E. Charbon, E. Felt and A. Sangiovanni-Vincentelli, "Automation of IC Layout with Analog Constraints", *IEEE Transactions on Computer-Aided-Design of Integrated Circuits and Systems*, vol. 15, pp. 923-942, August 1996.

[13] S. Rubin, *Computer Aids for VLSI Design*, Appendix B, Addison-Wesley, 1987.

[14] *Virtuoso Layout Editor User Guide*, Version 4.4.6, Cadence Design Systems Incorporated, 2000.

[15] J. K. Ousterhout, "Corner Stitching: A Data-Structuring Technique for VLSI Layout Tools", *IEEE Transactions on Computer-Aided-Design of Integrated Circuits and Systems*, pp. 87-100, January 1984.

[16] D. Marple, M. Smulders and H. Hegen, "Tailor: A Layout System Based on Trapezoidal Corner Stitching", *IEEE Transactions on Computer-Aided-Design of Integrated Circuits and Systems*, vol. 9, no. 1, pp. 66–90, January 1990.

[17] J. K. Ousterhout, G. T. Hamachi, R. N. Mayo, W. S. Scott and G. S. Taylor, "Magic: A VLSI Layout System", *Proceedings of IEEE/ACM Design Automation Conference*, pp. 152-159, June 1984.

[18] W. S. Scott and J. K. Ousterhout, "Magic's Circuit Extractor", *Proceedings of IEEE/ACM Design Automation Conference*, pp. 286-292, June 1985.

[19] S. L. Lin and J. Allen, "Minplex – A Compactor that Minimizes the Bounding Rectangle and Individual Rectangles in a Layout", *Proceedings of IEEE/ACM Design Automation Conference*, pp. 123-130, June 1986.

[20] Y. Bourai and C. J. R. Shi, "Symmetry Detection for Automatic Analog Layout Recycling", *Proceedings of Asian and South Pacific Design Automation Conference*, pp. 5-8, January 1999.

[21] D. Luenberger, *Linear and Nonlinear Programming 2nd Edition*, Addison-Wesley, 1984.

[22] D. Boyer, "Symbolic Layout Compaction Review", *Proceedings of IEEE/ACM Design Automation Conference*, pp. 383-389, June 1988.

[23] C. Bamji and R. Varadarajan, *Leaf Cell and Hierarchical Compaction Techniques*, Kluwer Academic Publishers, 1997.

[24] R. Okuda, T. Sato, H. Onodera and K. Tamaru, "An Efficient Algorithm for Layout Compaction Problem with Symmetry Constraints", *Proceedings of International Conference on Computer-Aided-Design*, pp. 148-151, November 1989.

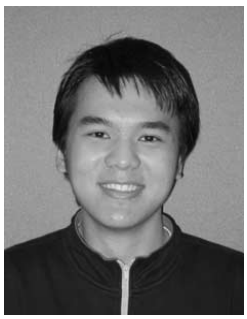
[25] T. H. Cormen, C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, MIT Press, 1990.

[26] N. Sherwani, *Algorithms for VLSI Physical Design Automation*, Kluwer Academic Publishers, 1999.

[27] G. Lakhani and R. Varadarajan, "A Wire-Length Minimization Algorithm for Circuit Layout Compaction", *Proceedings of International Symposium on Circuits and Systems*, pp. 276-279, May 1987.

Nuttorn Jangkrajarng received the B.Eng. degree in Electrical Engineering from Chulalongkorn University, Bangkok, Thailand, in 1997, and the MSEE. degree in Electrical Engineering from University of Washington, Seattle, WA, in 1999. He has joined the Mixed-Signal CAD Research Laboratory at the University of Washington and is currently working toward his Ph.D. degree. His research interests are in the field of mixed-signal VLSI and analog IC design automation.

Sambuddha Bhattacharya received his B.Eng. in Electrical Engineering from Birla Institute of Technology and Science, Pilani, India, in 1997. He received his M.S. in Electrical Engineering from the University of Washington, Seattle, in 2002. He has held position as an application engineer with Synopsys, India, and worked on placement, routing and timing convergence issues in digital design. He is currently working towards his Ph.D. degree in Electrical Engineering at the University of Washington. His research interests are in analog design automation techniques and timing and noise issues in digital circuits.



Roy Hartono received the B.S. in Electrical Engineering from University of Washington in 2002. He is currently pursuing the M.S. in Electrical Engineering from University of Washington. Since 2002, he has been joining the Mixed-Signal CAD Research Laboratory at the University of Washington. His interests are in VLSI design and automation.

C.-J. Richard Shi (M'91-SM'99) is currently an Associate Professor in Electrical Engineering at the University of Washington. His research interests include several aspects of the computer-aided design and test of integrated circuits and systems, with particular emphasis on analog/mixed-signal and deep-submicron circuit modeling, simulation and design automation.

Dr. Shi is a key contributor to IEEE std 1076.1-1999 (VHDL-AMS) language standard for the description and simulation of mixed-signal circuits and systems. He founded IEEE International Workshop on Behavioral Modeling and Simulation (BMAS) in 1997, and has served on the technical program committees of several international conferences. Dr. Shi has authored or co-authored over 100 papers published in international journals and conferences, and has served as the principal investigator of several research projects supported by DARPA, SRC and NSF with over $8M funding.

Dr. Shi received a Best Paper Award from the IEEE/ACM Design Automation Conference, a Best Paper Award from the IEEE VLSI Test Symposium, a National Science Foundation CAREER Award, and a Doctoral Prize from the Natural Science and Engineering Research Council of Canada. He has been an Associate Editor, as well as a Guest Editor, of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II, ANALOG AND DIGITAL SIGNAL PROCESSING. He is currently an Associate Editor of IEEE Transactions on Computer-Aided Design of Integrate Circuits and Systems.