Template-Driven Parasitic-Aware Optimization of Analog Integrated Circuit Layouts

Sambuddha Bhattacharya, Nuttorn Jangkrajarng and C-J. Richard Shi
Department of Electrical Engineering, University of Washington
Seattle, WA, 98195-2500
{sbb, njangkra, cjshi}@ee.washington.edu

ABSTRACT
Layout parasitics have great impact on analog circuit performance. This paper presents an algorithm for explicit parasitic control during layout retargeting of analog integrated circuits. In order to ensure desired circuit performance, bounds on layout parasitics' magnitudes are determined first. Then, graph techniques are coupled with mathematical programming to constrain layout geometry based on these parasitic bounds. The algorithm has been demonstrated to ensure desired circuit performance during technology migration and performance specification changes.

Categories and Subject Descriptors

General Terms: Algorithms, Performance, Design.

Keywords: Analog Layout Automation, Parasitics, Sensitivity, Optimization.

1. INTRODUCTION
Layout symmetry, device floorplans, relative placement and layout parasitics are of immense importance in ensuring desired analog circuit performance [1]. Layout parasitics arise from the transistor source/drain capacitances, interconnect resistances, and line and coupling capacitances. These parasitics can have significant impact on circuit performances such as gain, bandwidth, and phase margin. However, they cannot be accurately estimated before a layout is actually completed. This presents a major challenge to analog layout automation [2].

Recently, automatic template-based layout generation is emerging as an effective solution to analog layout automation. In the template-based analog layout automation tool IPRAIL (Intellectual Property Reuse-based Analog IC Layout) [3], an optimized layout is automatically generated from a symbolic structural template that contains device floorplan, symmetry, matching, and wiring alignment information. The templates can be extracted automatically from a coarse-grained layout generated by macro-cell based methods [2] or from an existing fine-tuned silicon-proven layout manually crafted by designers [3]. By automatically extracting and reusing the designers’ knowledge embedded in an existing layout, IPRAIL has demonstrated its efficacy for technology migration and electrical performance specification changes.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DAC 2005, June 13–17, 2005, Anaheim, California, USA.
Copyright 2005 ACM 1-59593-058-2/05/0006...$5.00.

In this paper, we present how template-based techniques can be explored for parasitic-aware optimization of analog circuit layout. A key observation is that the structural templates define the layout geometry in terms of constrained layout variables, therefore permitting the accurate modeling of layout parasitics parametrically in terms of these layout variables. Then constraints on layout parasitics can be enforced during layout generation.

With this, parasitic-aware analog layout generation is solved in three steps: (i) Identification of limits on parasitic values that guarantees desired performance, using the sensitivity of circuit performance to parasitics. (ii) Determination of constraints on layout geometric variables due to the parasitic bounds. (iii) Generation of target layout in presence of geometric constraints due to parasitics, symmetry, relative placement and design rules by combining graph-based methods and linear programming (LP).

2. PROBLEM DEFINITION

2.1 Layout Retargeting
Layout retargeting refers to the generation of a target layout from an existing layout [3]. In this method, a set of constraints corresponding to technology design rules, layout symmetry etc. is first extracted from the input layout. These constraints force the target layout to retain floorplan, symmetry and other properties of the existing layout. The objective of retargeting is to generate the target layout with minimum area while obeying these constraints.

Layout retargeting can be formulated as a one-dimensional compaction problem [3]. If $x_l$ and $x_r$ are the right and left ends of a layout, the problem in horizontal direction is given as:

$$\min \quad (x_r - x_l)$$

subject to $x_l - x_r \geq \text{const}$, $x_l - x_r = \text{const}$

where all variables correspond to the left and right edges of the layout rectangles. The constraints in Eqs. (1.2) and (1.3) correspond to design rules, fixed device widths and symmetry.

2.2. Parasitic Aware Layout Retargeting
Parasitic aware layout retargeting refers to the generation of a target layout from an existing layout such that the layout parasitics in the target layout are maintained within acceptable limits. These restrictions on parasitic values impose geometric constraints. The interconnect parasitics can be estimated with simple expressions for resistance and capacitance given as $r = \frac{1}{wL}$ and $c = 2LC_{SW} + \frac{wL}{CSW}$ where $R_{SJ}$ is the sheet resistance, $C_{SW}$ and $C_{S}$ are the sidewall and area capacitances and $L$ and $w$ are the length and width of the wire. The coupling capacitances are given as a linear function of $1/d$ where $d$ is the distance between two wires.

Parasitic aware layout retargeting refers to the problem defined in Eq. (1) with additional geometric constraints due to parasitics. Clearly, the parasitic models are nonlinear in terms of geometry and depend on both the horizontal and vertical dimensions of the wires. However, for superior computational speed, the geometric
3.1. Extraction of Input Layout Parasitics

Fig. 2: Flow of parasitic dictated geometric constraints generation.

3. PARASITIC CONSTRAINT GENERATION

The steps for computing geometric constraints due to parasitics are shown in Fig. 2. First, parasitics are extracted from the input layout. The sensitivity of the circuit performance parameters to the parasitics is computed next. Circuit sensitivities, parasitics, and the performance parameters are related by a set of linear constraints. We formulate a geometric programming (GP) problem with these constraints to maximize the bounds on the values of parasitics. These bounds are then mapped to the layout geometric constraints.

3.2. Sensitivity-Based Bounds for Parasitics

From the maximum allowable deviations of the performance parameters and the sensitivity of the performance parameters to the parasitics, a set of linear constraints are generated as follows.

\[ \sum_{j=1}^{N_p} S_y^j P_j \leq \Delta F_{\text{max}} \quad \forall F_i \geq F_i^{\text{nom}} \]  

(8)

\[ \sum_{j=1}^{N_p} S_y^j P_j \leq \Delta F_{\text{max}} \quad \forall F_i \leq F_i^{\text{nom}} \]  

(9)

where \( S_y^j = S_y^j \) if \( S_y^j \geq 0 \) and \( S_y^j = 0 \) if \( S_y^j < 0 \). \( S_y^j = -S_y^j \) if \( S_y^j \leq 0 \) and \( S_y^j = 0 \) if \( S_y^j > 0 \).

Larger parasitic bounds result in geometric constraints that are easier to be satisfied during layout generation. This computation of bounds on the parasitics is modeled as a GP problem of the form

\[ \min \ P_i^{\alpha_i} \]  

(10)

subject to the linear constraints in Eq. (8) and (9). Here, the \( \alpha_i \) are positive constant weights given according to the relative magnitude of the corresponding parasitics in the input layout.

3.3. Linearized Geometric Constraints Generation

Once the bounds on each individual parasitic value are obtained, they need to be translated to geometric constraints on each section of the wires. Consider the interconnect wire shown in Fig. 3(a). Let \( R_{IB} \) and \( C_{IB} \) be the maximum resistance and capacitance of its leftmost section. Then the following equations relate the width and lengths of the section to the parasitic bounds.

\[ x_2 - x_1 \leq \left( \frac{R_{IB}}{f_{\text{sh}}} \right) (y_4 - y_3) \]  

(11)

\[ x_2 - x_1 \leq \frac{C_{IB}}{2} \left( 2C_{IB} + C_d \right) (y_4 - y_3) \]  

(12)

The width and length of the section and the bounds on the sections resistance and capacitance define a parasitically feasible region shown shaded in Fig. 3(b). In addition, there is a range of length and width of the wire that would allow the target layout to be constructed. This geometrically feasible region is illustrated with the constraints due to parasitics need to be linearized in both horizontal and vertical dimensions to the form in Eq. (2).

\[ x_i - x_j \geq \text{const} \]  

\[ x_i - x_j \leq \text{const} \]  

(2.1)

(2.2)

Eq. (2.1) refers to geometric lower and upper bounds due to parasitics. Eq. (2.2) arises for matched interconnects.

2.3 Methodology

We incorporated our algorithm into IPRAIL. The steps in constraint generation are shown in Fig. 1(a). First, all transistors, passive devices and nets are extracted from the input layout. The design rule, connectivity, coupling and symmetry constraints are extracted next. Determination of bounds on parasitic values and generation of geometric constraints are described in Section 3.
dotted area. Our objective is to define the geometric upper bounds due to parasitics so as to maximize the overlap between geometrically feasible and parasitic feasible regions.

![Fig. 3: (a) Sections of a wire with current directions. (b) The feasible region defined by the resistive and capacitive bounds for the leftmost horizontal section of the wire is shaded. The dotted region represents the corresponding geometrically feasible region.](image)

Initial geometric feasible region for each wire section in the horizontal direction is obtained by two runs of longest path algorithm on the constraint graph, from the left to the right and from the right to the left. At this stage, the constraint graph does not include the parasitic constraints. The minimum widths are estimated based on the locations of the layout rectangles.

The generation of constraints on dimensions of wires requires fitting the largest rectangle in the overlap regions of geometric and parasitic feasibility. Let \( L_i \) and \( W_i \) be the lengths and widths of the \( i \)-th wire section. Let \( R_g \) and \( C_g \) be the bounds on the resistance and capacitance of the corresponding net and \( L_{i\_\text{min}} \) and \( W_{i\_\text{min}} \) be the geometrically feasible minimum sizes. Then the maximum overlap between geometric and parasitic feasible regions can be formulated as the following GP problem and solved with the optimization library of [6].

\[
\begin{align*}
\min & \quad \sum (L_i, W_i)^T \\
\text{subject to:} & \quad \sum L_i/W_i < R_g \\
& \quad \sum (C_i, L_i, W_i + 2 c_{SW} \cdot L_i) < C_g \\
& \quad L_i > L_{i\_\text{min}}, \quad W_i > W_{i\_\text{min}}
\end{align*}
\]

4. LAYOUT GENERATION

After the generation of the geometric upper and lower bound constraints due to parasitics, a horizontal and a vertical constraint graph are constructed with constraints due to design rules, connectivity, symmetry and parasitics. Here, we explain our algorithm with the horizontal constraint graph. If all the horizontal constraints are feasible, then the longest path algorithm can be employed directly to solve the compaction problem [3]. Upon its completion, the longest path algorithm finds the x-positions of the left and right edges of all layout rectangles. However, some constraints due to the parasitics may be infeasible because of two reasons. First, the geometric feasible region for a wire section may change due to parasitics in other layout rectangles. Second, the bounds on the parasitics may be too tight to be accomplished.

Geometric upper bounds due to parasitics of the form of Eq. (2.1) show up as negative weight arcs from the node for the right rectangle edge to the node for the left rectangle edge. These arcs, along with design rule left-to-right arcs can produce cycles. Positive cycles occur for small negative arc weights that render the sum of all arc weights of the cycle greater than zero. The longest path algorithm fails to terminate in presence of positive cycles as the layout dimension increases to infinity.

The resolution of positive cycles from the constraint graph is essential for the modified compaction problem. Resolution refers to reassignment of the constraint weights such that the sum of all arc weights is zero or less. We employ a combination of longest path and LP based constraint refinement to solve this problem. First, the algorithm identifies the positive cycles during a longest path run and then extracts the constraints due to parasitics in the positive cycle. It then resolves the positive cycle by refining the constraints due to those parasitics. The algorithm is shown in Fig. 4. Here, care needs to be taken so that resolution of one positive cycle does not introduce other positive cycles in the graph. This is based on the following observation.

\textit{Obs.:} Increasing the magnitude of a negative arc weight does not introduce a positive cycle in other parts of the constraint graph.

We increase the magnitude of the negative arc weights corresponding to parasitics in order to resolve the positive cycles. This amounts to increasing the geometric upper bound due to parasitics. We proceed by resolving one positive cycle at a time by reassigning parasitic-dictated geometric constraints. This is accomplished by formulating an LP problem.

![Fig. 4: Algorithm for layout generation with parasitic constraints.](image)

Consider a positive cycle in the horizontal constraint graph with \( N \) negative weight arcs due to parasitics. Let \( T_i \) represent the variable corresponding to the \( i \)-th negative weight arc. Let \( R_i \) and \( C_i \) be the resistance and capacitance of the corresponding wire section. The reassignment of arc weights can be formulated as the following LP problem.

\[
\begin{align*}
\min & \quad \sum \alpha R_i + \beta C_i \\
\text{subject to:} & \quad \sum T_i > \text{Positive\_cycle\_weight} \\
& \quad T_i < K_i R_i \\
& \quad T_i < L_i C_i
\end{align*}
\]

where \( T_i, R_i \) and \( C_i \) are the decision variables while \( \alpha, \beta, K_i = \frac{1}{(2c_{SW} + c_A \cdot y_{\text{Width}})} \) are constants.

The constraint graph is updated with the new weights obtained upon optimization and the algorithm is applied iteratively until all positive cycles are resolved. After the resolution of all positive cycles, the longest path algorithm settles to find the exact positions and sizes of all layout rectangles in the horizontal direction. The algorithm is then applied on the vertical constraint graph.

After the longest path algorithm settles in both directions, the parasitic values in some wire sections may still be above the bounds. This can be refined by relaxing these parasitic bounds by allocating some ‘unused’ parasitics from other wire sections.

5. RESULTS

The algorithm has been incorporated into IPRAIL. We present the results of parasitic-driven retargeting on a two-stage Miller-compensated operational amplifier (opamp) shown in Fig. 5 and a single-ended folded cascode opamp shown in Fig. 6. The opamps were designed initially in TSMC 0.25um CMOS technology and retargeted to TSMC 0.18um with new specifications.

The parasitics that affect circuit performances are indicated in Fig. 5 and Fig. 6. The bounds on the parasitic resistance and capacitance
were obtained from sensitivity-based optimization and are listed in the 2nd columns of Table 1. With these bounding values for the parasitics, the target layouts for the two-stage and folded-cascode opamps were obtained through parasitic-aware retargeting (PAR). The layouts were also retargeted without parasitic considerations (RWOP) for comparison. Parasitic values extracted from the target layouts for the respective cases are shown in Table 1. Here, the resistance values include metal, contact and gate-poly resistance. For multi-terminal nets, we report the sum of the parasitics of the resistance values.

The layouts were also retargeted without parasitic considerations (RWOP) for comparison. Parasitic values extracted from the target layouts for the respective cases are shown in Table 1. Here, the resistance values include metal, contact and gate-poly resistance. For multi-terminal nets, we report the sum of the parasitics of the resistance values. Table 1 shows the parasitic bounds obtained from sensitivity analysis, and parasitic values measured from layouts obtained by PAR and RWOP. The two-stage design shows significant differences in the parasitic values measured from layouts obtained by PAR and RWOP. The compensation zero is pushed inside the unity-gain frequency and leads to stability issues.

The simulation results for the two designs obtained with netlists called IPRAIL. This has been employed to retarget several analog layouts across technologies in a few seconds of CPU time.

Table 3 reports the statistics on the number of constraint graph nodes, number of design rule, symmetry and parasitic arcs for the two-stage and cascode opamps for PAR and RWOP. The two-stage opamp requires 12 positive cycle resolutions after generation of initial geometric constraints due to parasitics. The cascode opamp requires 4 positive cycle resolutions in PAR. For both designs, the target layout is generated within 15 seconds of CPU time.

6. CONCLUSIONS

In this paper, we presented an algorithm that enables explicit parasitic control in template-based retargeting of analog layouts. Bounds on layout parasitic values are obtained based on sensitivity analysis of circuit performance. Geometric programming is then employed to map the bounds on parasitic values to constraints on layout geometry. The target layout is then generated by an iterative longest path method with potential refinement of the parasitic dictated geometric constraint through linear programming. The algorithm has been incorporated into a computer-aided design tool called IPRAIL. This has been employed to retarget several analog layouts across technologies in a few seconds of CPU time.

REFERENCES


Table 3: PAR and RWOP statistics for two-stage and cascode opamps.

<table>
<thead>
<tr>
<th></th>
<th>Two-stage Opamp</th>
<th>Cascode Opamp</th>
</tr>
</thead>
<tbody>
<tr>
<td># Nodes in Constraint-Graph</td>
<td>PAR</td>
<td>RWOP</td>
</tr>
<tr>
<td>Design Rule, Sym Constraints</td>
<td>7598</td>
<td>9914</td>
</tr>
<tr>
<td>Parasitic Constraints Arcs</td>
<td>176</td>
<td>0</td>
</tr>
<tr>
<td>Pos. Cycles Resolved - hor /ver</td>
<td>2 / 10</td>
<td>-</td>
</tr>
<tr>
<td>Template Extraction Runtime</td>
<td>4.9 s</td>
<td>5.9 s</td>
</tr>
<tr>
<td>Layout Generation Runtime</td>
<td>10.1 s</td>
<td>4.6 s</td>
</tr>
</tbody>
</table>

The two-stage layout obtained by PAR is shown in Fig. 8. Table 3 reports the statistics on the number of constraint graph nodes, number of design rule, symmetry and parasitic arcs for the two-stage and cascode opamps for PAR and RWOP. The two-stage opamp requires 12 positive cycle resolutions after generation of initial geometric constraints due to parasitics. The cascode opamp requires 4 positive cycle resolutions in PAR. For both designs, the target layout is generated within 15 seconds of CPU time.