OPTIMUM STACKED LAYOUT FOR ANALOG CMOS ICs

Enrico Malavasi Davide Pandini Valentino Liberali†

Dipartimento di Elettronica ed Informatica
Università di Padova, Italy

†Dipartimento di Elettronica
Università di Pavia, Italy

ABSTRACT

A rigorous and efficient technique is presented for module generation in a maximally stacked layout paradigm for CMOS analog integrated circuits. Analog constraints on symmetry and matching provide a key for heuristics substantially reducing the computational complexity of robust graph algorithms. The solution found minimizes a cost function accounting for parasitic control and routability considerations. Combined with sensitivity analysis and automatic constraint generation, this algorithm provides a suitable performance-driven approach to analog layout module generation. Examples are reported showing the effectiveness of our approach.

1. INTRODUCTION

In recent years, several approaches to the automatic synthesis of analog integrated circuits have been proposed [1, 2, 3]. Significant efforts have been made toward a consistent performance-driven methodology [4], such that the respect of high-level specifications is guaranteed in all design stages. However, a severe discontinuity is present between schematic definition and physical implementation. Most of the existing tools for high-level architectural selection and circuit sizing are based on numeric optimization [5, 6]. Aim of these approaches is to produce schematics with transistor sizes and component values accounting for high-level performance and process specifications. Unfortunately, some performances such as phase margin and bandwidth are strongly influenced by circuit parasitics, which can be evaluated only after the layout is completed. Therefore, optimization cannot account for these performances unless considerations on layout are included.

So far no methodology has been proposed to automatically account for layout in early stages of circuit design. In [7] a design style is proposed, aiming to a *full-stacked* layout paradigm with densely abutted transistor modules. Advantages of this style are small stray capacitances, easy characterization, improved routability and minimal area. In [8] a layout-driven performance optimization approach is described, suitable for operational amplifiers. However, the designer is required to provide the detailed floorplan of the layout, and the result relies heavily on the user's expertise.

In this paper we propose a methodology for automatic generation of optimally stacked layouts in CMOS analog circuits. The optimality criterion is the minimization of a cost function weighing area, critical junction capacitances and routing length. Parasitic criticality is computed with a sensitivity analysis of the electrical performances on the circuit. Bounds on all junction capacitances are defined by a constraint generator [9] based on high-level performance specifications. In our algorithm, transistors are split into *modules* following a set of rules based on a pattern recognition approach. Modules are then abutted to form stacks by a *chaining algorithm*, which partitions the circuit graph into chains corresponding to the transistor stacks. The partitioning problem is known to be NP-complete. However, analog constraints are the key for an original branch-and-bound heuristic substantially reducing the computational cost of the chaining algorithm.

This paper is organized as follows. Section 2 provides an overview of our stack generation approach. In Section 3 the module splitting procedure is described. Our chaining algorithm is illustrated in Section 4, and in Section 5 the cost function driving the automatic stack generation is described. The branch-and-bound heuristic and the predictor on which it is based are described in Section 6. Experimental results are reported in Section 7.

2. AUTOMATIC STACK GENERATION

Given a CMOS circuit $\mathcal{C}$, its *circuit graph* $G = \{V, E\}$ is defined as follows: its vertices are the electrical nodes of the circuit, and an edge links two vertices if the associated nodes are respectively the source and drain of a transistor in the circuit. A *chain* is a connected subgraph whose nodes have either one or two adjacent edges. Since chains can be represented as ordered sets of edges, set operators such as '$\bigcup$', '$\bigcap$' and '$\in$', can be applied to them. An n-chain is a chain containing n edges. A stack of n transistors in $\mathcal{C}$ can be created if the corresponding edges in G form an n-chain. Every organization in stacks of the layout corresponds to a *chain partition* of the circuit graph, namely a set $\mathcal{P}$ of chains such that:

$$p \bigcap q = \phi \qquad \forall p, q \in \mathcal{P} \tag{1}$$

$$\bigcup_{p \in \mathcal{P}} p = E \tag{2}$$

where E is the set of edges in G. Condition (1) is the *non-overlapping* condition that no two stacks in the layout contain the same transistor. Condition (2) is the *covering* condition, that each transistor must appear in a stack. Notice that in every circuit at least one trivial partition $\mathcal{P}_0$ exists, where each chain has exactly one edge. This is often the starting configuration adopted by abutment-based placement tools [3, 10].

Stack generation follows a four-step procedure:

Step 1 The circuit graph is split into k subgraphs G_i, $i = 1, \ldots, k$ each containing only transistors with the same bulk bias node.

Step 2 Module splitting is performed on all transistors, according to topological adjacency relations and matching constraints.

Step 3 Each subgraph is split into maximal subgraphs containing only device modules with the same width.

Step 4 A *chaining algorithm* is applied to each subgraph to determine its best (i.e. the one minimizing the cost function) partition into chains.

Stack generation is followed by a post-processing step, where all specifications are checked against the constraints. Geometric constraints on size and aspect ratio are enforced by splitting large stacks and applying step 4 again. Matching constraints and parasitic bounds are enforced by abutting separate stacks by means of dummy modules.

The role of steps 1 and 3 is to reduce the cost of the algorithm, since complexity is exponential with the graph size. Their implementation is straightforward. Hence only steps 2 and 4 will be described in detail in the following sections.

3. MODULE SPLITTING

Transistors with big W/L ratio are implemented using two or more devices in parallel, all with the same length but shorter width. The strategy adopted for automatic module splitting is based on a set of pattern recognition rules. Each rule detects peculiar configurations of transistors and selects the devices to be split. The configurations recognized by our rules are the following.

1. Differential pairs, matched cascode stages, current mirrors.
2. Groups of two or more transistors with matching constraints.
3. Groups of two or more transistors with no matching constraints, but topologically adjacent.
4. Transistors whose width exceeds either an user-specified upper bound, or the maximum circuit size.

Specific module splitting can be explicitly required for one or more devices. Pattern-matching rules are applied sequentially in the order patterns have been enumerated. Once a transistor is split, the new modules are considered as matched devices, each introducing a new edge into the circuit graph. The remaining rules are applied to the modified circuit where the new modules take the place of the split transistor. In the case of rule 4, the large transistor width is divided by the smallest integer ensuring the respect of the geometric constraint. In the other cases, the width of the new modules is the Greatest Common Divider (GCD) of the widths of all the transistors involved. A lower bound applies, based on layout rules and user-specified minimum device width. Rules are not applied if the GCD is smaller than this bound.

4. THE CHAINING ALGORITHM

Step 4 is implemented by an improved version of the chaining algorithm described in [11]. First all possible chains in the circuit are generated by a dynamic programming procedure. Then the problem of finding a chain partition is transformed into a *clique* problem [12, p.194] and solved with the Bron-Kerbosch algorithm [13]. Every chain is a node for a *chain-graph* G_c, whose edges link two nodes if and only if the corresponding chains are *mutually compatible*, that is if they can coexist in the same partition. At each step of the algorithm, an attempt is made to augment a strongly connected subgraph (or "strong component"), If augmentation is not possible, the strong component is maximal, i.e. it is a clique. Since the non-overlapping condition (1) is necessary for mutual compatibility, every partition is a clique in G_c. Every clique found is checked against the coverage condition (2) to determine whether it constitutes a legal partition.

In [11] a *pairing criterion* was proposed to apply this algorithm to a layout paradigm, first described in [14], where n-type and p-type transistors are respectively aligned along parallel rows. However, in analog circuits n-type and p-type transistors are not usually balanced in number and size as they are in standard CMOS logic gates. Also, the order in which different devices appear in a stack is relevant, since it affects parasitics, matching and symmetry constraints. Finally, because of large transconductances, noise constraints and matching requirements, transistors with large W/L ratios are often needed. Hence, module splitting yields configurations where several devices are connected in parallel. Therefore, the number of edges is often greater than the number of vertices, and the size of the clique problem becomes huge even with small circuits. As an example consider the simple case of the differential pair shown in Figure 1. If each transistor is split into 5 modules, the circuit graph has 3 nodes and 10 edges. If all edges were considered as distinct devices in parallel, G_c would contain 65675 nodes.

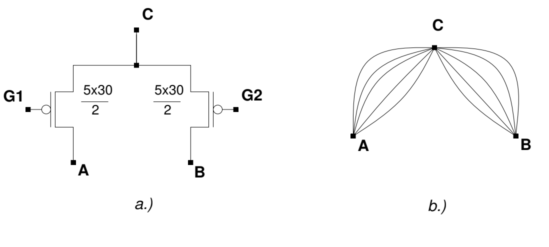

Figure 1: a.) A differential pair with transistors split into five modules. b.) Its circuit graph G.

By exploiting all analog constraints such as matchings, symmetries and parasitic bounds, we have achieved a substantial computational speedup. In our algorithm we have introduced the following modifications.

- Chains containing the same edges in different orders are kept distinct. However, edges corresponding to modules of the same transistor are mutually interchangeable. In the example of Figure 1, this reduces the number of different chains to 84.
- Junction capacitances are different for diffusions in internal and lateral positions in a stack. Alternative implementations are weighted by a cost function described in Section 5, on the ground of their impact on circuit specifications.
- Symmetry constraints introduce additional necessary conditions for mutual compatibility of chains. An edge links two nodes in G_c only if the chains involved can coexist in the same partition without necessarily violating symmetry constraints. By reducing the number of edges of graph G_c, symmetries effectively decrease the size of the clique problem.
- Precise matching can be enforced by requiring device modules to have the same current direction if they lay within the same stack.

5. THE COST FUNCTION

The junction capacitance of the diffusion regions located on an end of a stack is generally larger than the capacitance of internal regions [15, pp.129ff.]. For each module width W we denote by $C_{ext}(W)$ and $C_{int}(W)$ the maximum values of junction capacitances for diffusion regions respectively in external and internal positions in the stack. Their values can be either accurately computed with SPICE simulations at the circuit operating point, or they can be upper bounds derived from a worst-case analysis.

Let n_j be an electrical node, connected to M_j transistor modules. We denote by $C(n_j)$ its total capacitance towards the substrate. If all M_j modules have the same width W, the minimum possible value $C^{(min)}(n_j)$ of $C(n_j)$ is:

$$C^{(min)}(n_j) = \begin{cases} \dfrac{M_j}{2} C_{int}(W) & \text{if } M_j \text{ is even} \\ \dfrac{M_j - 1}{2} C_{int}(W) + C_{ext}(W) & \text{if } M_j \text{ is odd.} \end{cases} \tag{3}$$

If the modules have different widths, $C^{(min)}(n_j)$ is the sum of the capacitances computed with expression (3) for each width. The maximum possible value $C^{(max)}(n_j)$ of $C(n_j)$ is the capacitance due to M_j diffusions in external positions:

$$C^{(max)}(n_j) = \sum_{k=1}^{M_j} C_{ext}(W_k)$$

where W_k is the width of the k-th module. These limits are used by the constraint generator PARCAR [9] to generate bounds $C^{(b)}$ for all capacitances, in such a way that $C^{(min)}(n_j) \leq C^{(b)}(n_j) \leq C^{(max)}(n_j)$, and all circuit specifications are met provided all capacitances are below their bounds. Bounds are used to define a set of *criticality weights* for stray capacitances:

$$w(n_j) = \left[\frac{C^{(max)}(n_j) - C^{(b)}(n_j)}{C^{(max)}(n_j) - C^{(min)}(n_j)} \right]^2 . \tag{4}$$

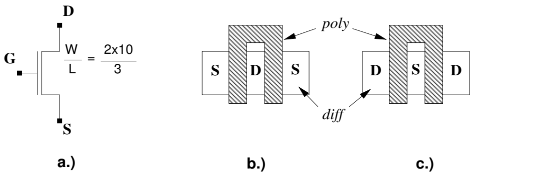

Figure 2: a.) A transistor split in two modules. b.) Layout minimizing the capacitance of node D. c.) Layout minimizing the capacitance of node S.

All criticality weights are between 0 and 1. Weights are close to 1 for capacitances with tight bounds ($C^{(b)} \approx C^{(min)}$). They are close to 0 for capacitances with loose bounds ($C^{(b)} \approx C^{(max)}$). The quadratic dependence of $w(n_j)$ on $C^{(b)}(n_j)$ matches the quadratic flexibility model used by PARCAR to generate the bounds.

Let s_i be a stack containing M_i modules of width W_i. Its $M_i + 1$ diffusion regions are connected to $M_i + 1$ electric nodes $n_j, \quad j = 1, \ldots, M_i + 1$. The cost of stack s_i is defined as follows:

$$F_s(s_i) = \sum_{j=1}^{M_i+1} k_j(W_i) \cdot w(n_j). \tag{5}$$

The *position weights* k_j account for the different junction capacitances due to the node positions in the stack. If $j = 1$ or $j = M_i + 1$, then $k_j = C_{ext}(W_i)/C_{int}(W_i)$, otherwise $k_j = 1$. The cost of a partition is defined as the sum of the costs of all its stacks.

As an example, consider the 2-module transistor shown in Figure 2.a and its two implementations 2.b and 2.c. Suppose the criticality weights for nodes S and D are respectively $w(S) = 0.8$ and $w(D) = 0.4$. Let us assume $C_{ext}/C_{int} = 1.6$. The costs of the two stacks are:

$$F(s_i) = \begin{cases} 1.6 \cdot 0.8 + 1 \cdot 0.4 + 1.6 \cdot 0.8 = 2.96 & \text{(layout 2.b)} \\ 1.6 \cdot 0.4 + 1 \cdot 0.8 + 1.6 \cdot 0.4 = 2.08 & \text{(layout 2.c)} \end{cases}$$

Solution 2.c is 30% cheaper than solution 2.b. The choice between partitions with the same cost can be taken based on area or matching criteria. If area is a critical issue, routability is optimized by selecting the solution with minimum component spread. Otherwise, matching is optimized by solutions with maximum device interleaving. This strategy automatically generates common-centroid structures when symmetry constraints are taken into account.

6. THE BRANCH-AND-BOUND HEURISTIC

Branch-and-bound is applied to the strong component augmenting step of the Bron-Kerbosch algorithm. Let F be the cost of the cheapest clique which can be yielded by augmentation of a strong component of G_c. If an estimate $\tilde{F}$ of F can be provided, it is possible to decide whether augmentation is worthwhile. If $\tilde{F}$ is larger than the cost of the cheapest clique found so far, augmenting is not carried out and the component is discarded. In order to guarantee that all cheapest cliques are found (*admissibility* condition), $\tilde{F}$ must be an optimistic estimate, i.e. it must be $\tilde{F} \le F$.

In what follows we will denote by N the total number of modules in the circuit. Let $s_i, \quad i = 1, \ldots, M_c$ be the stacks corresponding to the M_c chains of a strong component of G_c. The estimate has the following expression:

$$\tilde{F} = \sum_{i=1}^{M_c} F_s(s_i) + \tilde{h}$$

where $F_s(s_i)$ is the cost of stack s_i given by expression (5) and $\tilde{h}$ is an estimate of the cheapest combination of chains needed to complete a clique.

Let M be the number of modules in all the stacks of the strong component. By the covering condition (2), exactly $(N-M)$ edges are needed to augment the strong component to a clique. For sake of simplicity suppose that all the missing modules have the same width W. The cheapest arrangement for them is one stack with $(N - M - 1)$ nodes in internal positions and 2 nodes in external position. Let M_j be the number of modules connected to node n_j. The expression of $\tilde{h}$ is the following:

$$\tilde{h} = \sum_{j=1}^{(N-M+1)} \tilde{k}_j \cdot w(n_j). \tag{6}$$

where $\tilde{k}_j$ is an optimistic estimate of k_j. The value of $\tilde{k}_j$ is $M_j/2$ if M_j is even, it is $(M_j - 1)/2 + C_{ext}(W)/C_{int}(W)$ if M_j is odd. The predictor has no information about node positions in the stack. Therefore, expression (6) corresponds to the stack cost definition (5), with all position weights equal to 1, except the ones corresponding to nodes connected to an odd number of modules. In fact such nodes must be connected at least to one diffusion region in an external position. By construction this estimate is optimistic and the branch-and-bound heuristic satisfies the admissibility condition.

The generalization to the case in which modules have different widths is straightforward. The cheapest implementation is a layout with one stack for each value of width and $\tilde{h}$ is the sum of the costs of such stacks, each computed using expression (6).

7. IMPLEMENTATION AND RESULTS

The algorithm described in this paper has been implemented in about 4000 lines of C code. In what follows, CPU times are referred to a SparcStation-2 IPX.

The OTA folded cascode of Figure 3.a was constrained by specifications on unity-gain bandwidth, phase margin and low-frequency gain, and by a set of symmetry constraints. The largest chain-graph after Step 3 has 1756 nodes and 50191 edges. On this graph, the unconstrained clique problem is so large it couldn't be solved with the computing resources available to us, as all storage capabilities were exceeded after several hours of computation. With symmetry and matching constraints, it was reduced to 426 nodes and 899 edges and solved with the branch-and-bound heuristic in 2.6 CPU seconds. Each of the 8 solutions found contains three 10-chains. In particular, the partition shown in Figure 3, with maximum module interleaving, is the same obtained by hand by an experienced designer.

The OTA shown in Figure 4.a is a more complex example, where the bias generator is to be integrated with the amplifier. More than 2500 minimum-cost partitions were found in 8 seconds of CPU. One of them, with maximum

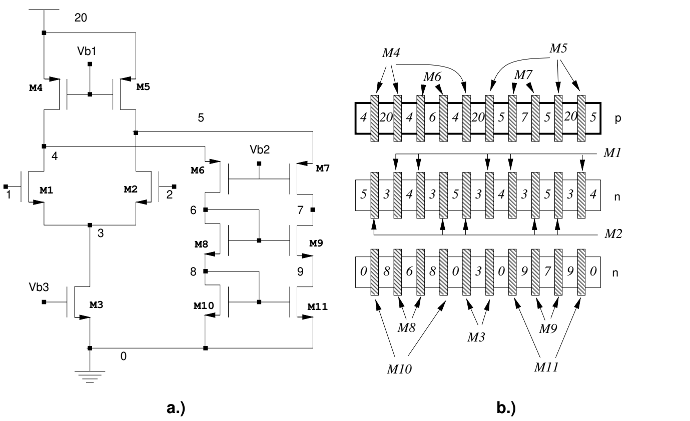

Figure 3: a.) Schematic of an OTA. b.) One of the 8 partitions found, equal to the hand-made solution.

matched-component spread and common centroid placement for the differential pair, is shown in Figure 4.b.

In all our tests, the branch-and-bound algorithm based on analog constraints has proven very effective in keeping the clique problem within reasonable size. So far, the range of practical applications of our approach has not been limited by the non-polynomial complexity of the clique solver.

8. CONCLUSIONS

An algorithm for maximally stacked layouts in CMOS analog circuits has been presented. With standard graph techniques, analog constraints on symmetry and matching provide a key for heuristics substantially reducing the computational complexity of the algorithms used. The approach always finds a solution minimizing a cost function weighing all junction capacitances. Combined with sensitivity analysis and automatic constraint generation, this algorithm provides a suitable performance-driven approach to analog layout module generation.

Future research will be aimed to the development of a layout-driven optimization methodology for circuit sizing. The algorithm described in this paper can be effectively used to evaluate the quality of layout implementation for automatic circuit design.

ACKNOWLEDGEMENTS

The authors would like to thank André Slenter (Philips Research), John Cohn (IBM Corp.), and Edoardo Charbon (University of California, Berkeley), for many discussions and for their expert advice on layout and CAD topics.

REFERENCES

[1] M. G. R. Degrauwe et al., "Towards an Analog System Design Environment", *IEEE Journal of Solid State Circuits*, 24:3, pp. 659–671, June 1989.

[2] S. W. Mehranfar, "A Technology-Independent Approach to Custom Analog Cell Generation", *IEEE Journal of Solid State Circuits*, 26:3, pp. 386–393, March 1991.

[3] 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*, 26:3, pp. 330–342, March 1991.

[4] H. Chang et al., "A Top-down, Constraint-Driven Design Methodology for Analog Integrated Circuits", in *Proc. CICC*, pp. 841–846, May 1992.

[5] R. Harjani, R. A. Rutenbar and L. R. Carley, "Analog Circuit Synthesis for Performance in OASYS", in *Proc. ICCAD*, pp. 492–495, November 1988.

[6] H. Y. Koh, C. H. Séquin and P. R. Gray, "OPASYN: A compiler for CMOS operational amplifiers", *IEEE Trans. on CAD*, 9:2, pp. 113–126, February 1990.

[7] U. Gatti, F. Maloberti and V. Liberali, "Full Stacked Layout of Analogue Cells", in *Proc. ISCAS*, pp. 1123–1126, 1989.

[8] H. Onodera, H. Kanbara and K. Tamaru, "Operational-Amplifier Compilation with Performance Optimization", *IEEE Journal of Solid State Circuits*, 25:2, pp. 466–473, April 1990.

[9] U. Choudhury and A. Sangiovanni-Vincentelli, "Constraint Generation for Routing Analog Circuits", in *Proc. DAC*, pp. 561–566, June 1990.

[10] E. Charbon, E. Malavasi, U. Choudhury, A. Casotto and A. Sangiovanni-Vincentelli, "A Constraint-Driven Placement Methodology for Analog Integrated Circuits", in *Proc. CICC*, pp. 2821–2824, May 1992.

[11] S. Wimer, R. Y. Pinter and J. A. Feldman, "Optimal Chaining of CMOS Transistors in a Functional Cell", *IEEE Trans. on CAD*, CAD-6:5, pp. 795–801, September 1987.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability*, W. H. Freeman & Co., New York, 1979.

[13] C. Bron and J. Kerbosch, "Algorithm 457 - Finding all cliques of an undirected graph", *Comm. ACM*, 16:9, pp. 575–577, Sep.1973.

[14] T. Uehara and W. M. vanCleemput, "Optimal Layout of CMOS Functional Arrays", *IEEE Trans. on Computers*, C-30:5, pp. 305–312, May 1980.

[15] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, Addison Wesley, Reading, MA, 1985.

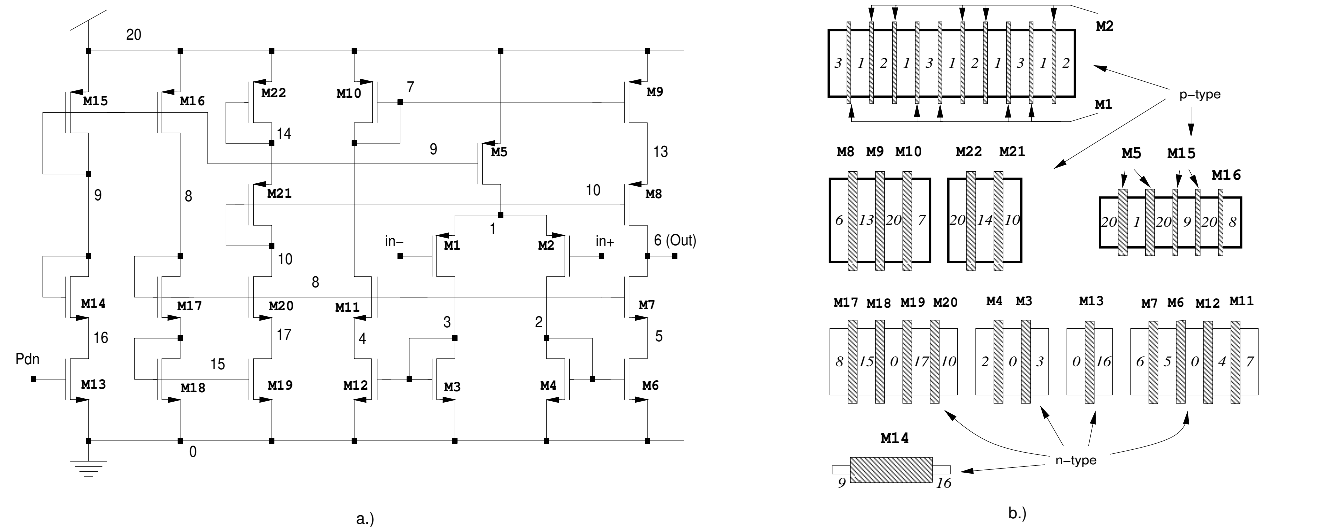

Figure 4: Schematic of an OTA. b.) One of the partitions found.