A Matching-based Placement and Routing System for Analog Design

Po-Hung Lin[†‡], Ho-Che Yu[‡], Tian-Hau Tsai[‡], and Shyh-Chang Lin[‡]
Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan[†]
RD Center, Springsoft Inc. Hsinchu 300, Taiwan[‡]

Abstract

Matching placement and routing is very important in layout design of high performance analog circuits. This paper presents a matching-based placement and routing system for custom layout design automation especially for analog or mixed-signal designs. The system explores various device-level matching-placement and matching-routing patterns to generate the most compact and high-quality layouts. Inputting a circuit netlist, the system automatically analyzes the circuit and extracts matching devices to form several matching device groups. Then, it selects the best matching placement and routing pattern for each device or device group to optimize and to meet the overall placement objectives and constraints. All patterns are user-configurable, stored in the pattern database, and portable from design to design. After the layout of each device and device group is generated and placed, the constraint-driven shape-based router is invoked to complete the layout. The overall system can easily generate high-quality layouts and greatly reduce the layout design time.

I. Introduction

Layout design automation has been researched for decades of years, and there has been a number of layout automation tools which have been popularly used in large-scale digital designs. However, layout for analog or mixed-signal circuits has been a manual, time-consuming, and error-prone task. There is almost no dominant layout automation methodologies or successful commercial tools in this area.

In the past decade, some automated layout generation methodologies are proposed in both academia and industry. Rutenbar et al. reported their survey on layout tools for analog ICs and mixed-signal SoCs in [11]. We simply classify the existing approaches into the following two categories:

- Template-driven layout
  This approach is commonly applied to module generation [1] or layout re-targeting [5]. The layout is generated based on a known layout pattern or layout template which specifies necessary device-to-device, device-to-wire, and wire-to-wire spatial relationships for a typical circuit. It is usually fast and easy to obtain compact layout. However, this approach lacks of flexibility if the layout designer wants to change the layout style from design to design.
- Constraint-driven layout
  It is more flexible than template-driven layout approach. Figure 1 shows the general flow of the constraint-driven or performance-driven layout [9]. It usually starts with the circuit analysis based on the netlist and/or performance specification of the design to generate the layout constraints. The placement and routing process is required to meet the constraints, and the final compaction stage is applied to optimize the area utilization. Although this approach is flexible and configurable, the layout quality is still not as good as the

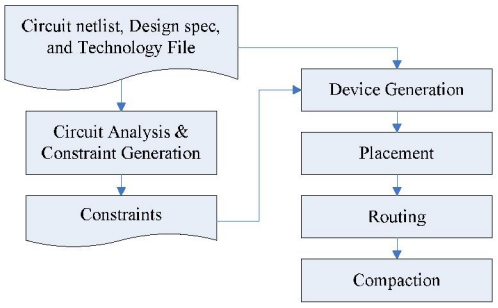

Fig. 1. The constraint-driven analog layout generation flow.

In this paper, a matching-based placement and routing system is proposed, and it has been embedded in the custom layout automation system, Laker©. The system applies constraint-driven layout generation flow and takes advantages of template-driven approach for device generation. It contains unified constraint database, circuit analyzer, circuit pattern database, matching placement and routing database, and constraint-driven placer and router. In addition, a high-level layout editor enabling matching pattern creation, modification and reuse gives designers the ability to perform faster and easier manual optimization without tedious work. The overall system can fast generate high-quality layouts and reduce the layout design time.

The remainder of this paper is organized as follows. Section II gives a system overview. Section III proposes the methods of matching device extraction. Section IV presents pattern-based device generation. Section V gives the ideas about placement-driven device generation and constraint-driven inter-device routing. Section VI reports the experimental results, and Section VII concludes this paper.

II. System Overview

Figure 2 shows the proposed matching-based placement and routing flow in the layout system. There are three databases in the proposed system including circuit pattern database, constraint database, and matching placement & routing pattern database. All of them are user-configurable. The circuit pattern database stores various circuit patterns that devices in these circuit patterns should be matched in the layout design. The constraint database stores the matching-device groups of a given circuit and general placement & routing constraints. The matching placement & routing pattern database stores different matching styles for the layout generation of each matching-device group.

Initially, the matching devices are extracted from the given circuit. The devices are formed several matching groups according

1-4244-0583-1/07/$20.00 ©2007 IEEE

to the circuit patterns in the circuit pattern database. Then, the placer will try to optimize the placement based on the given constraints and automatically generate the best device layout of each device or device group according to the matching placement and routing database. Finally, the constraint-driven inter-device router is invoked to complete the layout.

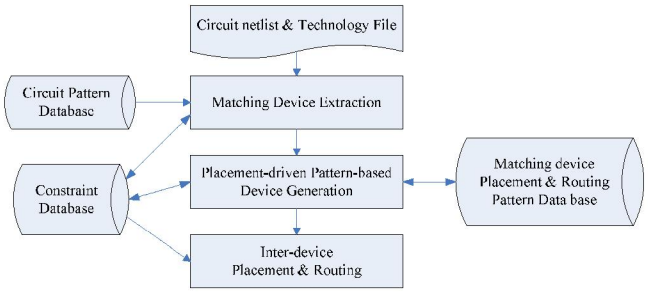

Fig. 2. The proposed matching-based placement and routing flow.

III. Matching Device Extraction

The circuit pattern database contains several kinds of circuit patterns that have to be placed and routed in matching style, such as current mirrors, differential pairs, or some symmetric devices [7]. Figure 3 shows three types of current mirror patterns in our circuit pattern database. In each type of the current mirror, device $M1$ should be matched with device $M2$. For the wide swing current mirror in Figure 3(c), device $M3$ should also be matched with device $M4$. There are more circuit patterns described in [6].

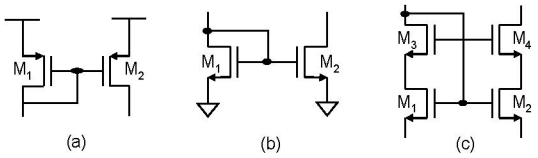

Fig. 3. Three types of current mirror. (a) Load current source. (b) Tail current source. (c)Wide swing current mirror.

Inputting a circuit netlist, the system automatically extracts these matching devices by tracing current/signal flows to find symmetry device pairs and to recognize circuit patterns. The matching device extraction flow and its relationship with circuit pattern database and constraint database are shown in Figure 4. Initially, the circuit netlist are transformed into the bipartite graph. There are two types of nodes in the bipartite graph. One of the types is device-node and the other is signal-node. There is an edge between a device node and a signal node in the bipartite graph if the device is connected to the signal.

Before starting to analyze the circuit, the system checks if there is any pre-defined matching device groups in the constraint database. For example, if there is a matching device group in the constraint database, we have to make a group node for those matching devices in the bipartite graph to preserve the matching constraints in the constraint database. Once the pre-defined matching constraints are preserved in the graph, we can start to recognize more circuit patterns and incrementally update the constraint database until no more pattern can be found.

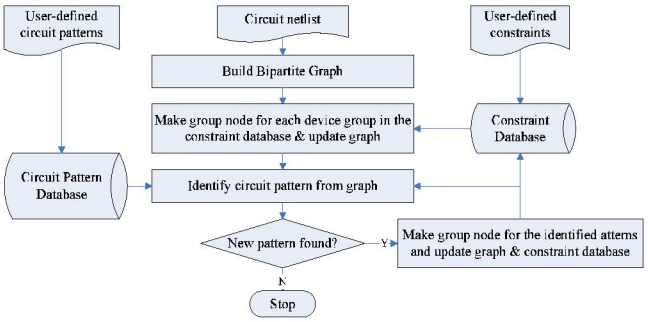

Fig. 4. The matching device extraction flow.

IV. Device Generation

The system applies the pattern-based matching placement and routing techniques to generate device layouts. There are commonly used matching placement and routing patterns which are predefined in the matching placement and routing pattern database. User can also customize their preferred patterns and add them to the pattern database. The system generates the layout of the matching devices according to user-specified pattern or based on the layout objectives.

In the following sub-sections, we will first introduce the pattern-based matching placement followed by pattern-based matching routing.

A. Pattern-based Matching Placement

There are several kinds of matching styles for matching placement, such as common centroid, symmetry, ... etc. The system pre-defines various matching patterns in the matching pattern database according to the number of the matching devices, the matching styles, the m-factor of each device, and the connectivity of of the matching devices. Figure 5 shows the pattern list for a matching group with two devices. The pattern named AAAAA_BBBBB indicates that both devices are folded into five elements, and the first row contains the five elements device A while the second row contains the five elements device B. Another pattern named AB_BA_AB_BA indicates that both devices are folded into four elements, and there are four rows in the matching placement. Each row contains one element of both devices, and the placement sequence between the neighboring rows are different, which is known as the common centroid matching style. The matching placement of the matching devices are directly released by one of the patterns.

B. Pattern-based Matching Routing

After the matching placement is done for each matching device group, the nets in the matching device group can be routed in the mean time by applying pattern-based matching routing. There are three kinds of pre-defined routing styles including backbone, matching-backbone, and matching-cross styles in the matching routing pattern database which are shown in Figure 6.

The routing flow is shown in Figure 7. First of all, the matching nets should be analyzed. In the example of Figure 8, two matching-net pairs, $net2$, $net3$ and $net5$, $net6$, can be automatically extracted.

After the matching net analysis, the pins of a net or a matching-net pair are partitioned into several pin groups based on the analysis of applicable routing patterns. For example, in Figure 8, the pins of the matching-net pair ($net2$, $net3$) are divided into three

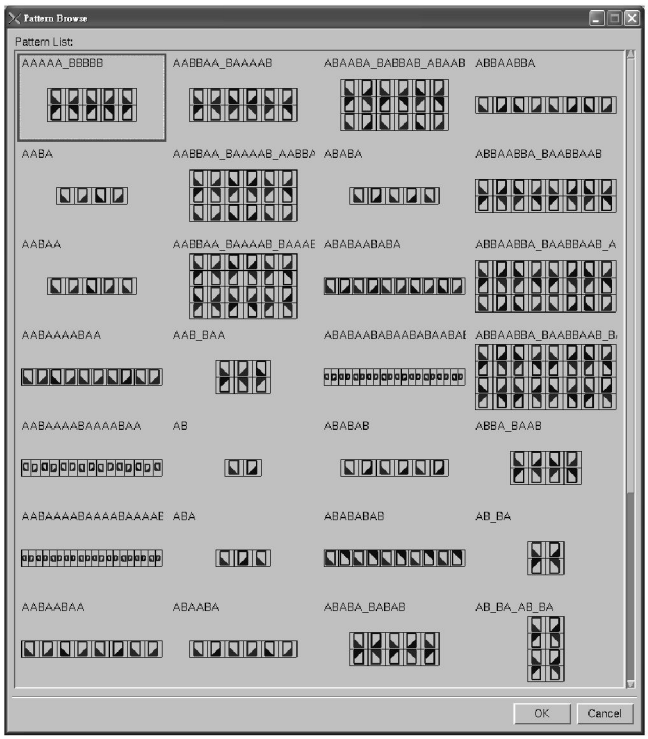

Fig. 5. Matching placement pattern list for a matching group with two devices.

pin groups and each are routed in matching-backbone style in different routing channels. The pins of another matching-net pair ($net5$, $net6$) are divided into two pin groups with matching-cross style in different channels. There is an additional virtual pin group with backbone routing style in the vertical channels to connect the wires of the same net among different horizontal channels. The net, $net4$, applies a special pattern which indicates the routing of the guard ring.

The routing sequence is determined by some pre-defined priorities in the following manner:

- A pin group of a matching-net pair has higher priority than that of a non-matching net.
- A pin group which can be routed in cross style has higher priority than a pin group which can only routed in backbone style.
- The routing in the horizontal channels has higher priority than that in the vertical channels.

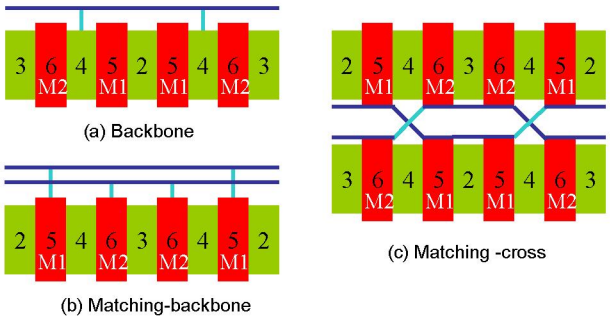

Fig. 6. Three routing styles. (a) Routing for a single net with backbone style. (b) Routing for matching nets with matching-backbone style. (c) Routing for matching nets with matching-cross style.

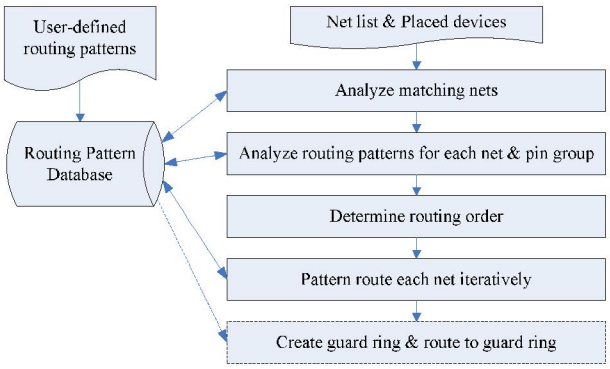

Fig. 7. Pattern-based device routing flow.

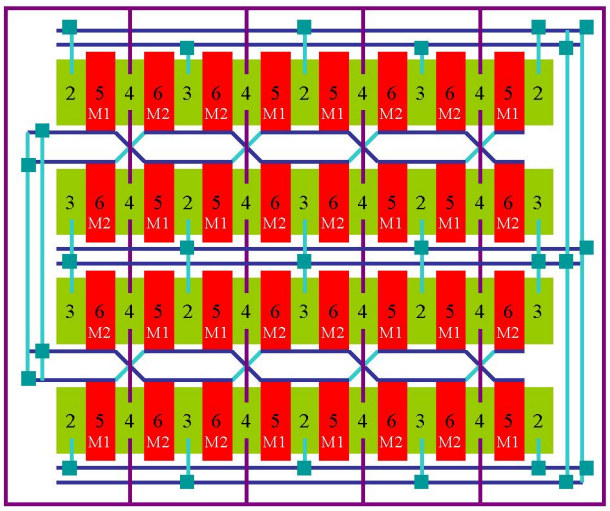

Fig. 8. A pattern-based placement and routing example.

- The guard-ring net is routed at the last.

Based on these routing priorities, the routing sequence of the case in Figure 8 is in the following order:

1. The cross-style routing of the matching-net pair ($net5$, $net6$) in the horizontal channels.
2. The matching-backbone routing of the matching-net pair ($net2$, $net3$) in the horizontal channels.
3. The matching-backbone routing of the both matching-net pairs ($net2$, $net3$) and ($net5$, $net6$) in the left and right vertical channel.
4. Finally, the guard-ring style routing of the net, $net4$.

V. Inter-device Placement and Routing

A. Inter-device Placement

When placing generated devices and matching groups, we apply the simulated annealing (SA) algorithm [8] based on the floorplan representations such as B*-trees [2] or sequence-pairs [10]. The objective of the placement is to minimize the placement area and the total wire-length while trying to meet the following placement constraints:

- Symmetry/alignment for some devices and/or device groups
- Preplacement and blockages

- Power/ground topology
- I/O pin locations
- Aspect ratio of the placement area

During the placement, the layout variant and matching pattern of each device or device group is automatically determined by the random perturbation operations in the SA process. To estimate the routing space between the devices or the device groups, we adopt the idea from KOAN [3], which adds the halo space for each generated device layout. The halo space is determined by the number of nets of the device, the available routing layers, and the design rules.

B. Inter-device Routing

The inter-device routing is implemented base on the tile-based gridless router [4]. We consider various routing constraints, which have been stored in the constraint database, including differential pair routing, equal length routing, net shielding, symmetry routing, ...etc. After completing the routing of the constraint nets in a pre-defined, or user-specified routing sequence, the unconstraint nets are then routed by the blockage-aware global router followed by the gridless detailed router. The final layout of the analog circuit can be obtained after the whole processes.

VI. Experimental Results

Figure 9 shows an example circuit of a CMOS operational amplifier. Nine matching device groups are automatically extracted. The placement-driven device generator generates the layout of each device and matching group by applying pattern-based matching placement and routing. In the mean time the dimension of the array-style devices are automatically determined during placement. Once the layout of each device and device group is generated and placed, the inter-device routing is performed. Figure 10 shows the automatically generated layout of the example circuit in Figure 9. The overall layout generation procedure based on our approach is within ten minutes, while an experienced layout engineer needs at least two days to complete the same layout.

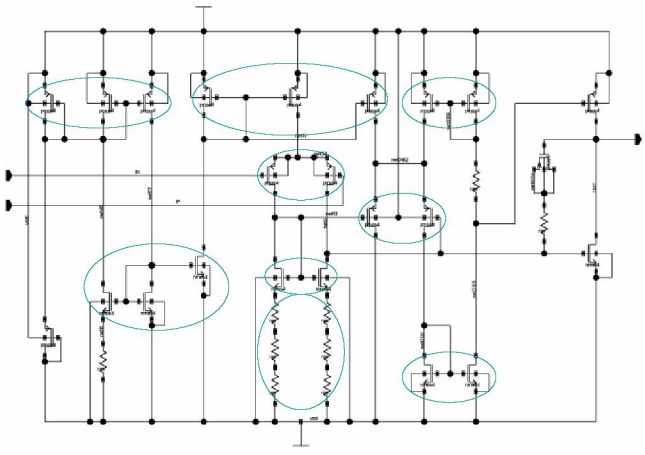

Fig. 9. An example circuit of a CMOS Op-Amp. Each circled device group forms a matching group.

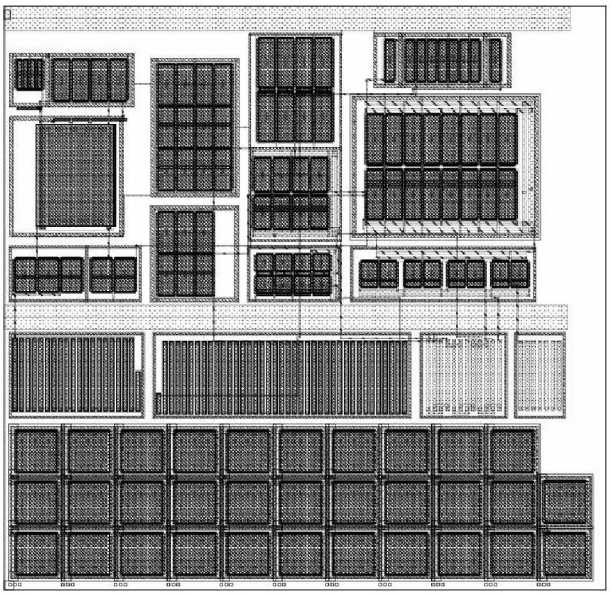

Fig. 10. The automatically generated layout of the Op-Amp in Figure 9.

VII. Conclusion

We have proposed a matching-based placement and routing system integrating matching device extraction, placement-driven pattern-based device generation, and shape-based routing technologies to facilitate custom layout generation. The three configurable and knowledgeable databases including circuit pattern database, constraint database, and matching device placement and routing database are helpful to guarantee layout quality from design to design. Experimental results show that our system can easily generate high-quality layouts and cut the layout design time from days to minutes.

References

[1] J. D. Bruce, H. W. Li, M. J. Dallabetta, and R. J. Baker, "Analog layout using ALAS!," *IEEE JSSC*, vol. 31, no. 2, pp. 271–274, Feb. 1996.

[2] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B*-Trees: a new representation for non-slicing floorplans," *Proc. DAC*, pp. 458–463, 2000.

[3] J. M. Cohn, D. J. Garrod, R. A. Rutenbar, and L. R. Charley, "KOAN/ANAGRAM II: New tools for device-level analog placement and routing," *IEEE JSSC*, vol. 26, pp. 330–342, Mar. 1991.

[4] J. Dion and L. M. Monier, "Contour: A tile-based gridless router," Western Research Laboratory Research Report, Palo Alto, California, 1995.

[5] G. J. Gadelkarim, T. Vucurevich, and W. H. Kao, "Circuit layout technique with template-driven placement using fuzzy logic," *U.S. Patent*, 5768479, Jun. 1998.

[6] H. Graeb, S. Zizala, J. Eckmueller, and K. Antreich, "The Sizing Rules Method for Analog Integrated Circuit Design," *Proc. ICCAD*, pp. 343–349, 2001.

[7] Q. Hao, S. Dong, S. Chen, X. Hong, Y. Su, and Z. Qu, "Constraints generation for analog circuit layout," *Proc. ICCCAS*, pp. 1339–1343, 2004.

[8] S. Kirkpatrick, C. D. Gelatt, and M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[9] D. Long, Y. Zeng, C. Du, X. Hong, and S. Dong, "A novel performance-driven automatic layout tool for analog circuit," *Proc. ICCCAS*, pp. 1344–1348, 2004.

[10] H. Murata, K. Fujiyoushi, S. Nakatake, and Y. Kajitani, "Rectangle-packing-based module placement," *Proc. ICCAD*, pp. 472–479, 1995.

[11] R. A. Rutenbar and J. M. Cohn, "Layout tools for analog ICs and mixed-signal SoCs: A survey," *Proc. ISPD*, pp. 76–83, 2000.