A GENERAL ROUTER FOR ANALOG LAYOUT

Enrico Malavasi, Michele Chilanti, Roberto Guerrieri
Dipartimento di Elettronica, Informatica e Sistemistica,
via Risorgimento, 2 - 40136 Bologna - ITALY

Abstract

This contribution presents a software tool for the automatic routing of analog layouts, which is able to deal with most problems associated with analog routing, such as symmetric circuit architectures, proximity constraints and control over the parasitic parameters. The routing strategy, based on a modified version of Lee's algorithm and the high-level user interface are presented. Results are reported with examples of circuits with peculiar architectures, correctly routed by our tool.

1 Introduction

Analog parts of complex systems are often integrated with digital circuitry on the same chip. Therefore the availability of a CAD environment, where tools for analog and digital circuit design coexist, is important to reduce the overall design time [1]. While many tools have already been produced for several steps of the digital synthesis (e.g. [2], [3] [4]), few can be found for the analog-circuit design: in [5] analog cells with fixed topology are synthesized following a case-specific rule-based approach which requires a substantial effort when the synthesis of new circuits has to be added to the system capabilities; the problem of analog routing is not specifically dealt with in [6], where a digital router is used in a module generator for CMOS operational amplifiers (op-amp); a system for automatic layout of analog cells is presented in [7], which though does not discuss a few problems of analog routing, such as the preservation of symmetries.

Analog layout synthesis is tied to the satisfaction of a set of design, technology and performance constraints [8, pp. 515-516] [1], whose relative weight can be so different for digital and analog circuits that often a good tool for the former is not suitable for the latter, and vice versa:

- component match: it directly influences the offset of op-amps and the quality of current mirrors;
- control over the maximum amount of capacitive coupling between signal paths and power, ground or clocks;
- control over the voltage drops due to undesired stray resistances;
- control over such peculiar architectural characteristics as symmetries [9] and their implementation in the layout architecture.

The routing is therefore complex, because many constraints are to be accounted for. The algorithm must look for a trade-off between the routing efficiency and the respect of the user's constraints.

In section 2 a maze-routing algorithm which can handle the complexity of the analog routing problem is introduced. In section 3 a tool implementing the described algorithm is presented, with the routing strategy adopted. In section 4 we illustrate how the user can specify the required constraints. The results follow in section 5, with some examples showing the capabilities of the tool.

2 The routing algorithm

The algorithm we propose is a modified Lee's multipoint maze-router on a three-dimensional relative grid with dynamic allocation [10]. This approach gives the greatest generality to the problem formulation and does not impose topology constraints. Furthermore, the control over the cost function may be as accurate as needed in order to exploit constraints stemming from interactions among wires. With the dynamic allocation of the grid, a large memory occupation is not necessary in order to achieve propagation efficiency. Wherever the grid is not properly refined, new grid nodes can be easily generated, without losing the control over the local net congestion.

A description of the basic Lee's algorithm can be found in [10] and [11].

The cost function

In our algorithm the cost function is associated with the edges between the grid nodes, rather than with the nodes themselves [12]. So doing we can

get different indexes depending on the direction from which a wave gets the node. The cost function takes into account the following elements:

- the weighted distance between nodes;
- a layer-resistivity parameter;
- a layer-to-bulk capacity parameter;
- a proximity parameter dependent on the distance of the node from the already existing wires;
- a congestion parameter, based on the real wire crowding of the surrounding area and on an estimate of its final crowding based on a fast first-attempt path-search for each of the remaining wires.

The weights for all these components are set by the user for each of the wires to be traced.

The backtrace

During the backtrace, we have to choose among the equivalent paths leading to the same final electrical connectivity, but giving rise to wires with different shapes. In this case the path minimizing the final number of corners is selected. At the end of the backtrace, an automatic routine builds up the wire using the given path. The new wire is allocated to the edges between the grid nodes, rather than to the nodes themselves. Wherever none of the adjacent edges is available, the grid nodes are split, new nodes are dynamically allocated in the grid and the edges linking them are used for the wires. Therefore the grid represents the whole space available for routing.

The pre-connected pins

There is a possibility that several pins, among those to be wired, be already connected. In analog circuits this is a very frequent situation, because we often deal with transistors whose size is not minimum, and whose structure is stacked and compound [13] [14]. Therefore each internal node may be available in many different locations of the component border. The existence of these internal connections is exploited whenever possible, ignoring the interactions between the waves generated by already short-cut pins; the useless connections between couples of equivalent pins are therefore avoided and the routing turns out to be more compact in area and connection length.

Figure 1. The routing procedure for symmetric circuits: [a] The left-side wires are built. [b] Each left-side wire is sources from one of the left-side wires. [c] The left- and right-side wires are linked with the link-wires. [d] The final wires are obtained by subtracting the unused path segments.
Routing a symmetric circuit

Our algorithm can route symmetric nets or partially symmetric nets, that is nets having a part which is symmetric with respect to a given axis and another part which is not. The only necessary requirement is the symmetry of the placement. Even the symmetry of pins is not necessary, because the algorithm automatically recognizes and routes the symmetric part of the net, leaving the remaining part to the ordinary routing algorithm described before.

Let a vertical symmetry axis be given, splitting the circuit into a left and a right part. The algorithm is the following:

1. At the beginning we deal only with the left part of the circuit. For each of the nets tied by the symmetry relation, we consider only the pins laying on the left side, and with the ordinary routing algorithm a wire connecting these pins in built (left-side wire) (Figure 1a).

2. When all the left-side wires have been built, they are mirrored onto the right side of the circuit, creating the set of the right-side wires, one for each of the symmetric net pairs (Figure 1b).

3. Each right-side wire is linked to the left-side wire with the same electric node identifier, and with all the pins of the net which have not been connected yet. A third wire is therefore built for each net, called the link wire (Figure 1c).

4. Each triplet, i.e. left, right and link part of a net, is then used as the path of the final complete wire, routing the corresponding net. The unused path segments, due to non-symmetric parts of the net, are discarded (Figure 1d).

The control over parasitics

The matching of the user's constraints over parasitics is controlled in two ways: Before routing, the cost function is properly modified in order to make it difficult to get crossing and proximity between nets whose coupling is critical; after routing, a check of the parasitic capacitive coupling is carried out, using a detailed physical model of the wire. Then, weak or strong modifications can be performed, as described in the next section.

3 The routing strategy

Two different phases are alternatively performed in order to produce the final routing. During the first phase, an initial scheduling of the wires to be connected is produced and the most promising order is chosen. The actual building of the first wire (or set of wires) is then performed by the maze-router described above. This procedure is then iterated for each wire to be routed.

We have dealt with a large amount of circuits, using different scheduling strategies, both automatic and hand-driven. In this section the scheduling algorithm will be described. Then, the rip-up and re-route phase will be presented.

The scheduling algorithm

When two or more wires have still to be routed, their priority list can be compiled by a scheduler, which can also alter the cost function distribution, in order to take into account the area congestion and the control over the parasitic parameters. It takes into consideration the already routed wires and an estimate of the paths of the unrouted ones. With a repeated call to the scheduler, the whole set of wires can be routed, using for each of them the maximum amount of knowledge on the system available at the moment of its routing. The algorithm can be summarized as follows:

1. A simplified routing of all the unrouted nets is performed, using a simplified standard cost function: for each net a first-attempt path is found, the corresponding congestion is computed and the most likely critical crossing points are marked.

2. If specific symmetries of the architecture of the system are required, the wires without this characteristic are discarded, in order to accomplish the symmetric part of the circuit as soon as possible.

3. User-defined scheduling and tight electric specifications, if any, are taken into account: therefore power ground and low-level signals, if still unrouted, are selected, discarding the other nets.

4. For each of the selected nets, a priority cost is computed, using the information on congestion and critical areas drawn in (1). The net with the highest cost, i.e. the one whose routing is the most critical for the whole system, is selected.

5. Given the paths resulting from (1), the cost function is computed and the selected net is routed.

The rip-up and re-route phase

Even a smart scheduler is not able to get a high-quality analog layout. It can control the cost function, but not the detailed wire structure. A check phase is therefore necessary at the end of the routing procedure, in order to face the problem of the nets routed out of the user's constraints on parasitics. First the system performs a set of weak modifications, trying to match the constraints. If this strategy fails, it tries to build a shield, following the user's specification described in the next section.

4 The high-level user interface

In order to explicit the requirements associated with each connection, it is possible to "annotate" each wire using a high-level language which is then translated by the user-interface in a suitable sequence of routing tasks. With this language it is therefore possible to require the fulfillment of those constraints which are peculiar of an analog layout, namely:

- Control over coupling between wires. The command PROXIMITY-CONTROL <WIRE-1> <WIRE-2> <APPROACH> [SHIELDING-APPROACH]
- The command PROXIMITY-CONTROL <WIRE-1> <WIRE-2> <APPROACH> [SHIELDING-APPROACH]
- The command PROXIMITY-CONTROL <WIRE-1> <WIRE-2> <APPROACH> [SHIELDING-APPROACH]
- The command PROXIMITY-CONTROL <WIRE-1> <WIRE-2> <APPROACH> [SHIELDING-APPROACH]
- The command PROXIMITY-CONTROL <WIRE-1> <WIRE-2> <APPROACH> [SHIELDING-APPROACH]
- The command PROXIMITY-CONTROL <WIRE-1> <WIRE-2> <APPROACH> [SHIELDING-APPROACH]
- The command PROXIMITY-CONTROL <WIRE-1> <WIRE-2> <APPROACH> [SHIELDING-APPROACH]
- The command PROXIMITY-CONTROL <WIRE-1> <WIRE-2> <APPROACH> [SHIELDING-APPROACH]
- The command PROXIMITY-CONTROL <WIRE-1> <WIRE-2> <APPROACH> [SHIELDING-APPROACH]

4.5. For each of the selected nets, a priority cost is computed, using the information on congestion and critical areas drawn in (1). The net with the highest cost, i.e. the one whose routing is the most critical for the whole system, is selected.
5 Results

The previous algorithm has been implemented in a program of about 9000 lines of C code. An interface consistent with the Oct Symbolic Policy [16] has been implemented. Compilation has been performed by Sparcs [17], a tool designed for digital circuits and not specifically tuned for analog requirements. Our tool was tested on several examples drawn from layouts placed by experienced designers.

Figure 3 shows the layout of a folded cascade op-amp automatically routed by our tool. This is a particularly compact placement, with a typical stacked architecture, whose routing was performed for a single metal technology. All critical nets have been traced using the metal layer with the exception of one cross-under in polysilicon, not used in the hand-made layout. The stacked architecture, with a large number of internally connected pins, has been extensively exploited in order to achieve a very compact layout.

Figure 4 shows the compacted layout of a double-ended differential amplifier. Even though detailed inspection reveals minor violations of the symmetry requirements, due to the compaction phase, the overall quality of the final layout is adequate for most purposes.

6 Conclusions

An algorithm for the routing of analog circuits and the program implementing it have been presented. The algorithm can deal with most of the routing problems of analog circuits, as demonstrated by the examples previously discussed, which show the suitability of this method for a large class of circuits whose peculiar architecture requires a special care in their routing. The preliminary results are very promising, since the quality of the automatically generated layouts is comparable with that of hand-made ones, made by expert designers.

Acknowledgements

The authors wish to express their thanks to the researchers of CSELT laboratories, Torino and Prof. F. Maloberti for their many suggestions about the synthesis of analog circuits.

References