AN AUTOMATIC LAYOUT DESIGN AID FOR ANALOG INTEGRATED CIRCUITS

Lihong Zhang, Ulrich Kleine

Institute for Electronics, Signal Processing and Communication (IESK)
Otto-von-Guericke University of Magdeburg
PO Box 4120, D-39016 Magdeburg, Germany
E-mail: {lihong, kleine}@iesk.et.uni-magdeburg.de

ABSTRACT

This paper presents an automatic layout design system for analog integrated circuits. It is tailored for analog circuit designers so that they can bring their special knowledge and experience into the synthesis process to create high quality layouts. Designers can write and maintain their own technology and application independent module generators for subcircuits in a module generator environment. Different from other systems, layout generation is based on optimal complex modules rather than simple single devices. A novel genetic placement approach with simulated annealing is developed. The minimum-Steiner-tree based global routing can be integrated into the placement procedure in order to improve the routability of placement solutions. The compaction based constructive detailed routing finally realizes the layout of the whole circuit. Several examples are given to demonstrate its design efficiency and usability. Experiments show that it is superior to the existing tools and can provide comparable results to manual layouts.

1. INTRODUCTION

The synthesis process of analog circuits normally starts with the selection of an appropriate circuit topology or an existing circuit for a similar application. Then designers iteratively optimize this circuit using knowledge-based circuit simulations to meet the desired specifications. Due to technology scaling, the electrical parameters of MOS transistors get worse for analog circuits and therefore, the design window, in which the design parameters of elements must fit, decreases. Hence, the amount of optimization will increase in the near future, especially for low voltage applications where the design window is even smaller. For a reliable optimization, the knowledge of electrical parameters caused by the layout (e.g. parasitic capacitance and resistance etc.) and the behavior of different topologies are required. Therefore, the layout alternatives are already necessary for high performance circuits during circuit design.

So far most of analog layouts are still hand-crafted by specialists which is a considerable time-consuming process. As a result the optimization is normally performed without a layout and parasitic elements are only roughly estimated during the design and optimization phase. Several tools have been developed to automate the generation of analog layouts [1]-[3] by more or less bypassing designers. Usually these tools apply a top down approach taking the already optimized circuit netlist as input and generating the layout for this circuit. In [3] this optimized netlist is automatically generated. If the layout does not meet the specifications the circuit optimizer can be executed again. Other analog design tools, which do not support layout generation, have been presented in [4] where the parasitic elements arising from the layout can only be estimated. Unlike these automated design tools, which have been developed mainly for system designers, an automation layout tool called ALADIN (Automatic Layout Design Aid for Analog Integrated Circuits) is currently being developed for analog experts who can bring their specific knowledge into the synthesis process to create high quality layouts.

This paper is organized as follows. Section 2 introduces the design flow within the tool ALADIN. Section 3 explains the main components of the system including the module generator, placer and routers. Especially the novel genetic placement approach with simulated annealing is described in more detail. The experimental results are given in Section 4. Finally the conclusion is drawn.

2. DESIGN FLOW WITHIN ALADIN

In Fig. 1 the structure of ALADIN is given. The entire system mainly consists of three components: DesignAssistant, Module Generator Environment and Technology Interface. The DesignAssistant is integrated in Cadence Design Framework II. It provides a graphical user interface to optimize an analog circuit from schematic to layout. According to circuit partitioning, executable module generators defined by designers are used to create the layout of each subcircuit. The placer and routers are
employed to compose the layout of the whole circuit. A precise estimation of the parasitic capacitance of the circuit is obtained by an extraction of layout and back annotated into the original schematic. The module generator environment allows designers to write parameterizable, technology and application independent modules as complex as possible [5]. The technology interface eases the input of description rules including design rules, electrical rules and capacitance sensitivity matrices [6].

The global routing is performed simultaneously along with the placement procedure followed by the detailed routing. After the layout generation, an extraction is performed and the parasitic elements are automatically back annotated into the schematic. The entire circuit is then simulated with a good parasitic estimation and the optimization loop begins by changing the parameters of the circuit (parameter optimization) and/or by redefining the capacitance sensitivity matrix (application optimization).

3. SYSTEM COMPONENTS

3.1 Module Generator

Due to analog constraints like matching requirement, it is usual to build more or less complex clusters of devices, hereafter called modules, which are parameterized for the processed circuit. In contrast to bipolar circuits, the geometrical parameters of MOS transistors can vary in a wide range for analog applications. Therefore, module generators must be available which provide optimal layout topologies even for that wide range of parameterization. In recent years, several module generators [8]-[10] have been presented. These generator concepts use either a procedural approach with an absolute [8] or relative [9] placement of objects or a graphical approach [10], which is called "design by example". In practice, the module creation is often limited to non optimal single device modules [2] and thus the placement and routing problem is increased by necessary analog constraints like neighborhood and symmetry conditions. But sub optimal solutions in the module generation step often can not be made up in the subsequent placement and routing stages.

In general, a module generator library is provided during analog layout synthesis, which normally can not be changed by designers because of the complexity. These libraries are maintained by CAD experts which are usually not analog experts. In contrast to that, in ALADIN the module generator environment providing a tailored description language MOGLAN [11] based on a relative placement of objects or a graphical approach [10], which is called "design by example". In practice, the module creation is often limited to non optimal single device modules [2] and thus the placement and routing problem is increased by necessary analog constraints like neighborhood and symmetry conditions. But sub optimal solutions in the module generation step often can not be made up in the subsequent placement and routing stages.

In general, a module generator library is provided during analog layout synthesis, which normally can not be changed by designers because of the complexity. These libraries are maintained by CAD experts which are usually not analog experts. In contrast to that, in ALADIN the module generator environment providing a tailored description language MOGLAN [11] based on a relative placement of objects or a graphical approach [10], which is called "design by example". In practice, the module creation is often limited to non optimal single device modules [2] and thus the placement and routing problem is increased by necessary analog constraints like neighborhood and symmetry conditions. But sub optimal solutions in the module generation step often can not be made up in the subsequent placement and routing stages.

MOGLAN enables designers to describe parameterizable modules hierarchically and design-rule independently. The MOGLAN description is translated into C++ code. After this it is compiled into an executable technology and application independent generator with a C++ function.
library. This language features loops, conditional statements and a set of simple functions to create and to wire primitive geometry without considering absolute coordinates. Moreover the design rules are automatically evaluated and fulfilled. If a rule cannot be fulfilled, an error message occurs.

3.2 Module Placer

The genetic algorithms (GAs) differ from the other stochastic search techniques by being able to encode and exploit past information efficiently during a search. This learning ability is the reason why this strategy is chosen in ALADIN to solve the analog module placement problem, which is NP-complete. The conventional chromosomal representation of the GAs is based on bit-string [12]. A flexible bit-matrix chromosomal representation is employed here, where a chromosome is obtained by rearranging the genes from the bit-string to a two-dimensional array. This divide-and-conquer technique allows the GA to generate new configurations faster without degrading its search efficiency.

Each module is represented by a row matrix, whose normal formulation is \{X_1, X_2, ..., X_N, Y_1, Y_2, ..., Y_N, O_1, O_2, O_3\}. All the elements in this matrix are 1 or 0. The binary expressions of the triple \(X_1, X_2, ..., X_N\), \(Y_1, Y_2, ..., Y_N\) and \(O_1, O_2, O_3\) respectively represent the \(X\)-, \(Y\)-dimensional positions and 8 possible orientations (i.e., transformation and reflection) of each module. Different modules are packed along columns. Thus a \(M\)-module configuration is described by a matrix with the dimension of \(M*(N+3)\). The applied bit numbers of \(X\)- and \(Y\)-coordinate should be determined as a trade-off between accuracy and search efficiency. So they are treated as variables which can be set by designers.

The crossover operation is based on two chosen parents with the random cut lines. The combination of the left-top & right-bottom parts in one parent and the right-top & left-bottom parts in the other parent produces an offspring. Because of the probably involved inversion operations, the homologous processing should be done during the crossover operation. The mutation operator inverses one bit at random in the bit-matrix, which provides a way to widen the range of genes in a population to ensure that no gene permanently disappears from the population. The inversion operator shuffles the arrangement of genes in a chromosome by exchanging its rows and columns randomly, which weakens the linkage among genes in a chromosome.

The placement problem can be categorized into two portions: relative placement and absolute placement. Although the absolute placement has the advantage to cover every possible topology theoretically, the great complexity of multi-dimensional configuration space in the placement problem prevents the search practically. So in our approach, an idea of cell slide is employed to convert an absolute placement to a relative placement. A slide function is applied which adapts the position of cells after they are absolutely placed. During the adaptation, overlaps among cells are avoided so that the relative position instead of absolute coordinates becomes the ultimate focus. Its application makes full use of the advantages of both the absolute and relative placements. It can cover all the possible topologies with a good search efficiency. One example of the module slide is depicted in Fig. 3. Usually a critical module, such as A, is set as fixed so that the rest of cells are placed radially in reference to it afterwards. Modules B, C and D are shifted consecutively in the arrow direction. The numbers inside module blocks are the shift sequence.

Fig. 3: One example of the module slide.

The cost function is quite important as it includes all the interesting considerations and controls the searching procedure. Our cost function consists of four parts, which are given in (1),

\[
C = (c_{cellarea} + \alpha_{Nwell} c_{Nwellarea} + \alpha_{Pwell} c_{Pwellarea} + \alpha_{A} c_{Aarea}) + \alpha_{D} c_{Darea} + \alpha_{sym} c_{sym} + \alpha_{overlap} c_{overlap} + \alpha_{com} c_{com}
\]

\(\alpha^*\) are the weight factors for the corresponding cost \(C^*\), which balance the importance of all the possible considerations according to different design requirements. The first is the area cost which is made up of the whole area and NWELL, PWELL region areas. They will make NWELL or PWELL region relatively concentrated with the whole area decreased. The second is the net-length cost, which will be described in more detail in the next section. The third is the size cost, which is used to control the shape of the final layout. The fourth is the overlap penalty cost. Because the slide function can be optionally employed, this penalty cost is only useful if the slide function is disabled during the algorithm process. The last is the penalty cost of symmetry constraints.
Based on the respective study on GAs and simulated annealing (SA), a novel genetic placement approach with simulated annealing is developed [13]. Because SA is a general methodology rather than a completely specified algorithm, the combination of GA and SA is managed in a natural way based on the principle of GA.

Algorithm GASA(T₀)
(M: the size of one generation. a, b: variable for members. T₀: the initial temperature. T: temperature.)

Begin
1 input module geometry and net-list;
2 initialize the first population randomly;
3 evaluate the fitness;
4 while not (stopCriterion())
5   while (innerLoopCriterion())
6     foreach (M * crossoverRate)
7       choose the first parent with roulette wheel selection;
8       choose the second parent randomly;
9       do crossover to generate one offspring;
10    endfor
11   endwhile
12   choose the best M members among the former members & new generated offsprings, and set them as the new generation;
13   foreach (M)
14     do mutation on the clone (a) of one member (b);
15     if (accept(Cost(a), Cost(b), T))
16       b = a;
17    endif
18   endfor
19   endwhile
20 update mutationRate and T;
21 endwhile
22 output the best member;
End

Fig. 4: Outline of GASA.

The mutation is a random change made to each member of the population with a relative small probability, which enables new features to be introduced into a population. It can be considered as the random move in the SA and controlled by temperature which follows a certain cooling schedule. By this way the random chromosomal change of the GA is greater in the initial phase and becomes less and less until the search converges finally. Since only one parameter is involved, the Cauchy cooling schedule is applied. The initial mutation rate is set as 0.85. The algorithm outline is depicted in Fig. 4. stopCriterion() makes the evolution process terminated if no improvement has been observed for a predefined number of consecutive generations or a fixed number of generations are over. innerLoopCriterion() manages a certain local equilibrium. accept() follows the normal Metropolis’ procedure.

### 3.3 Module Routers

Inspired by the traditional routing routines for digital circuits, two routing phases including the global routing and detailed routing are employed in ALADIN. Some specific techniques are developed on the basis of analog circuit applications. The global routing is integrated into the placement procedure in order to improve the accuracy of routing estimation. The compaction based constructive detailed routing is applied to generate the final layout based on the placement solution.

Four basic net-length estimators including half-perimeter, center-of-mass, complete graph and minimum spanning tree are provided. The accuracy of net-length estimation is crucial for the optimization because it may make the time-intensive late search phase pointless. All the above methods are based on Manhattan distance, which inevitably degrades the reliability of estimation. Because nets play a critical role in analog circuits, due to crosstalk etc, the fifth method – the minimum Steiner tree is developed, which is not only for the net-length estimation but also for the global routing. The global routing acts as a mediator between placement and detailed routing so that detailed routing can successfully use the output of placement in the view of global optimization. A weighted graph is used to model the routing regions. A rectilinear channel graph is formed by passing channels (or edges) through critical regions and forming vertices at their intersections. Finding a global route becomes equivalent to finding a optimal subtree (the minimum Steiner tree) in the routing pin graph which spans the terminal vertices. A Dijkstra shortest path algorithm is applied to solve this minimum Steiner tree problem.

The routing area estimation during the placement can not be performed accurately enough no matter how advanced a placement algorithm is. It is especially distinguished for analog circuits because special constraints are imposed for different nets. Therefore the routing area is usually overestimated for routability when a single-phase routing approach is employed. So a compaction-based constructive detailed routing algorithm is employed in ALADIN. For each module, the interconnections within the module are first wired densely around the module boundary with a ring router. Then this dense module is compacted towards others according to the position relationship extracted from the placement solution. The interconnections among the compacting module and reference modules are routed within a relatively small scope with a modified maze router.

### 4. RESULTS

In this paper three circuits are presented to evaluate the effectiveness of the ALADIN tool. They could be used as benchmark circuits. In Fig. 5(a) the schematic of a CMOS...
A high speed regenerative comparator is depicted. In the reset phase, the output voltages are short-circuited. The latch is in its unstable working point. In the regeneration phase, a small voltage difference at the inputs will drive the outputs to different levels. The partitioning of the modules is indicated by the shaded rectangles. Its fully symmetric layout is automatically generated within ALADIN that prevents clock feedthrough effects. The die photo of the comparator fabricated in 0.8 µm-technology is shown in Fig. 5(b). The measured performance data are: Offset voltage < 3mV, Propagation delay time < 5ns, Power supply voltage 5V, Power dissipation 7mW. Because of the clock feedthrough, the clock signal should keep a certain slope, for instance, the transition rate of the rising slope or falling slope should be greater than 0.3% according to the measurement result in our experiment. Otherwise the circuit cannot work properly.

In Fig. 6(a) the schematic of an one-stage operational amplifier is depicted. The partitioning of the modules is indicated by the shaded rectangles. Its layout is automatically generated within ALADIN and its die photo is shown in Fig. 5(b). The specification of this circuit is: Power supply voltage 5V, open loop gain 48dB, 0dB-bandwidth for 5pF load capacitance 0.53MHz, slew rate 0.48V/µs, phase margin 91° and power consumption 0.57mW.

In Fig. 7(a) the schematic of a two-stage operational amplifier is depicted. The partitioning of the modules is indicated by the shaded rectangles. Its layout is automatically generated within ALADIN and its die photo is shown in Fig. 7(b). The specification of this circuit is: Power supply voltage 5V, open loop gain 84dB, 0dB-bandwidth for 20pF load capacitance 1.479MHz, slew rate 0.95V/µs, phase margin 90.2° and power consumption 0.22mW.
5. SUMMARY

In this paper a layout automation tool for analog integrated circuits has been introduced. It provides the possibility that circuit designers could bring their special knowledge and experience into the synthesis process to create high quality layouts. The main components and applied techniques are described consecutively. Designers can write and maintain their own technology and application independent module generators for subcircuits in a module generator environment. A novel genetic placement approach with simulated annealing is developed. Apart from the implemented multiple net-length estimators, the minimum-Steiner-tree based global routing can be integrated into the placement procedure in order to improve the routability of placement solutions. The compaction based constructive detailed routing finally realizes the layout of the whole circuit. Experiments show that it is superior to the existing tools and can provide comparable results to manual layouts.

6. REFERENCES