2-1-2015

A framework for fine-grain synthesis optimization of operational amplifiers

Taher Essam

The American University in Cairo

School of Science and Engineering

A Framework for Fine-grain Synthesis Optimization of Operational Amplifiers

A Thesis Submitted to

Electronics and Communication Engineering Department

In partial fulfillment of the requirements for

the degree of Master of Arts/Science

By Taher Essam Ali Kourany

Under the supervision of:

Prof. Yehea Ismail, Dr. Emad Hegazi

January/2015

Cairo, Egypt

The American University in Cairo

School of Science and Engineering (SSE)

A Framework for Fine-grain Synthesis Optimization of Operational Amplifiers

A Thesis Submitted by

Taher Essam Ali Kourany

Submitted to Department of Electronics

January/2015

In partial fulfillment of the requirements for

The degree of Master of Science

has been approved by

_____

Thesis Supervisor

Affiliation:

Date _____

_____

Thesis first Reader

Affiliation:

Date _____

_____

Thesis Second Reader

Affiliation:

Date _____

_____

Department Chair

Date _____

_____

Dean of SSE

Date _____

To my Parents and Egypt,

You Mean the World to me.

A man ought to read
just as his inclination leads him;
for what he reads as a task
will do him a little good.

SAMUEL JOHNSON

Acknowledgment

First of all, I have to thank God for giving me power and patience to finish my master thesis.

I'd like to show my gratitude for my supervisor Prof. Yehea Ismail for the great opportunity that he gave to me. I'd like to thank him for his support, guidance and for the great effort that he exerted during my masters. He provided me with all the facilities that I need in my research.

I'd like to show my gratitude to my dear brother Ali Kotb and my beloved friends Hoda Ahmed, Nehal Hussein, Hazem Medhat, and Dalia Ahmed for their support and help. They helped me a lot in solving the problems I faced. I'd like to thank them for their advice and good spirit. We spent unforgettable time together.

I'd like to make a special acknowledgment to Eng. Soha Hamed for her guidance throughout the difficulties i experienced while working on my thesis.

I am thankful for all those who discouraged me, it taught me perseverance. It's because of them, I did it myself.

Abstract

OF THE THESIS OF

Taher Essam Ali Kourany for Master of Science

Major: Electronics and Communication Engineering

The American University in Cairo

Title: A Framework for Fine-grain Synthesis Optimization of Operational Amplifiers

Supervisor: Prof. Yehea Ismail, Dr. Emad Hegazi

This thesis presents a cell-level framework for Operational Amplifiers Synthesis (OASYN) coupling both circuit design and layout. For circuit design, the tool applies a corner-driven optimization, accounting for on-chip performance variations. By exploring the process, voltage, and temperature variations space, the tool extracts design worst case solution. The tool undergoes sensitivity analysis along with Pareto-optimality to achieve required specifications. For layout phase, OASYN generates a DRC proved automated layout based on a sized circuit-level description. Morata *et al.* (1996) introduced an elegant representation of block placement called sequence pair for general floorplans (SP). Like TCG and BSG, but unlike O-tree, B*tree, and CBL, SP is P-admissible. Unlike SP, TCG supports incremental update during operation and keeps the information of the boundary modules as well as their relative positions in the representation. Block placement algorithms that are based on SP use heuristic optimization algorithms, e.g., simulated annealing where generation of large number of sequence pairs are required. Therefore a fast algorithm is needed to generate sequence pairs after each solution perturbation. The thesis presents a new simple and efficient *O(n)* runtime algorithm for fast realization of incremental update for cost evaluation. The algorithm integrates sequence pair and transitive closure graph advantages into TCG-S* a superior topology update scheme which facilitates the search for optimum desired floorplan. Experiments show that TCG-S* is better than existing works in terms of area utilization and convergence speed. Routing-aware placement is implemented in OASYN, handling symmetry constraints, e.g.,

interdigitization, common centroid, along with congestion elimination and the enhancement of placement routability.

Table of Contents

List of Figures

List of Tables

1.Introduction

An analog system is typically characterized by a set of performance parameters used to quantify the properties of the circuit. Given a fixed topology, circuit synthesis is the process of determining numerical values for all components in the circuit such that the circuit conforms to a set of performance constraints. The pervasive trend in recent years is the integration of whole systems into single-chip. Analog circuitry is widely used in systems applications such as telecommunications and robotics, where analog interfaces to an external environment are coupled with digital signal processing systems. The demands for high performance CMOS analog circuits increased dramatically in recent years, especially for digital–analog interface circuits, due to the emergence of system-on-chip (SoC). **A**lthough analog circuits take up only a minor part of most ASIC's, their design time and cost is very important. Most of knowledge, effort, and time are spent in designing analog blocks of the chip since they are largely dominated by heuristics and experience needed to achieve required specifications.

Given a set of specification/requirements that describe the system to be realized, the selection of the optimal implementation comes mainly out of experience. Many digital parts of such chips can nowadays be synthesized rapidly and reliably using CAD tools developed for semicustom design methods such as gate arrays, standard cells, and macro cells. On the other hand, analog subsystems still need to be entirely handcrafted by a specialist, due to the high degree of nonlinearity and interdependence among design variables. Therefore, the design time and cost associated with dedicated analog interface components often constitute a bottleneck in semicustom design of mixed analog/digital systems. The growing scale of industry and the rapid advancement in integrated circuits technology have led to dramatic increase in physical design complexity. The need to tackle this complexity and comply with time-to-market has encouraged the wide use of the hierarchical design and IP modules for a faster convergence to the optimum design in terms of area and speed. Some analog components are replaced with their digital counterparts, which are successful to a great extent. However, there are limitations to replace all the

analog blocks and what was left are considered to be intellectually challenging. The success of the digital design ideas and tools against analog design and its domination over the majority of the industry, due to sophisticated accurate tools empowering design time-to-market, exposed the lack of comparable analog semi-custom tools.

For a top down, knowledge based approach, analog synthesis problem can be decomposed into two parts: firstly the synthesis of sized circuits from behavioral specifications and secondly the IC layout generation from these circuits. Design automation ideas from digital IC design have only recently begun to migrate into analog circuit design. In part, this reflects the inherent complexities of the analog design process. Outside of conventional analog/digital systems, there has recently been great interest in the design of parallel analog VLSI signal processing architectures. Hence, it is clear that CAD tools must be developed to cope with both the complexity of large-scale analog circuit designs, and with the requirement for rapid design times. In the digital domain, structured abstractions and hierarchy are commonplace, and are relied upon to make seemingly large synthesis tasks tractable by breaking them into smaller steps. Such abstractions and hierarchy do not currently play a central role in analog design. Some ideas from digital design methodologies, such as standard cell libraries and module generators, have recently been applied to analog design tasks. However, such techniques usually have several drawbacks, e.g., libraries allow the designer to make only crude tradeoffs among performance specifications, and they become obsolete rapidly in the face of technological evolution. The numerical circuit simulator SPICE is often used as a benchmark of comparison to determine the relative accuracy of alternative schemes for evaluating the performance of analog circuits.

1.1. Literature Review

Synthesis comprises two steps: topology selection and sizing. Topology selection means selecting the appropriate circuit topology from a library of topologies. Sizing consists of choosing appropriate transistor dimensions and biasing voltages to satisfy a given set of performance specifications. Topology selection has proven very difficult to automate due to its knowledge-intensive nature. Many attempts have been made in order to mimic the designer's expertise and knowledge into automation tools. There exists two approaches adopted in analog circuit synthesis: knowledge based approaches and optimization based approaches [20]. In the knowledge-based stream, the designer extracts design equations and integrates them into the tool to be reused for the same topology. In the optimization-based approach, the optimizer searches the design space for the circuit that satisfies certain constraints and minimizes certain objectives. The optimization-based approach was further divided into two approaches: equation-based optimization and simulation-based optimization. In the equation-based optimization, circuit evaluation is done through pre-derived equations for performance specifications, initially extracted by the designer or by symbolic analysis. In the simulation-based optimization, the specifications are directly measured from the output waveforms of a simulator. The simulation-based approach has two major advantages over the equation-based approach:

• Accurate simulation models are used instead of approximate equations

• No long preparatory effort to extract all the describing equations. Practically, the extraction may rely fully on the simulator capabilities.

In order to reduce this design effort, analog standard cell libraries can be used. However, since the circuits are then not tailored to their application, an optimum solution, with respect to power dissipation and area, is not obtained. Furthermore, such libraries, which typically have required more than 20 man years of design effort, very rapidly become obsolete due to technology evolution. Stochastic combinatorial optimization methods such as simulated annealing and genetic algorithms (GAs) require the computation of

performance parameters for a large number of circuit sizing alternatives. It is, therefore, beneficial to reduce the time associated with generating performance estimates.

Synthesis tools adopting approaches to equation based strategies have been implemented. IDAC: An Interactive Design Tool for Analog CMOS Circuits [1] was one of the earliest tools developed in analog design automation, where designer has to specify the technology, desired building-block specifications. In IDAC, designer selects from different topologies existing in the database. Other tools [2], [3], [6], [22], [23], [24], [25], [26], and [27] adopted the same approach.

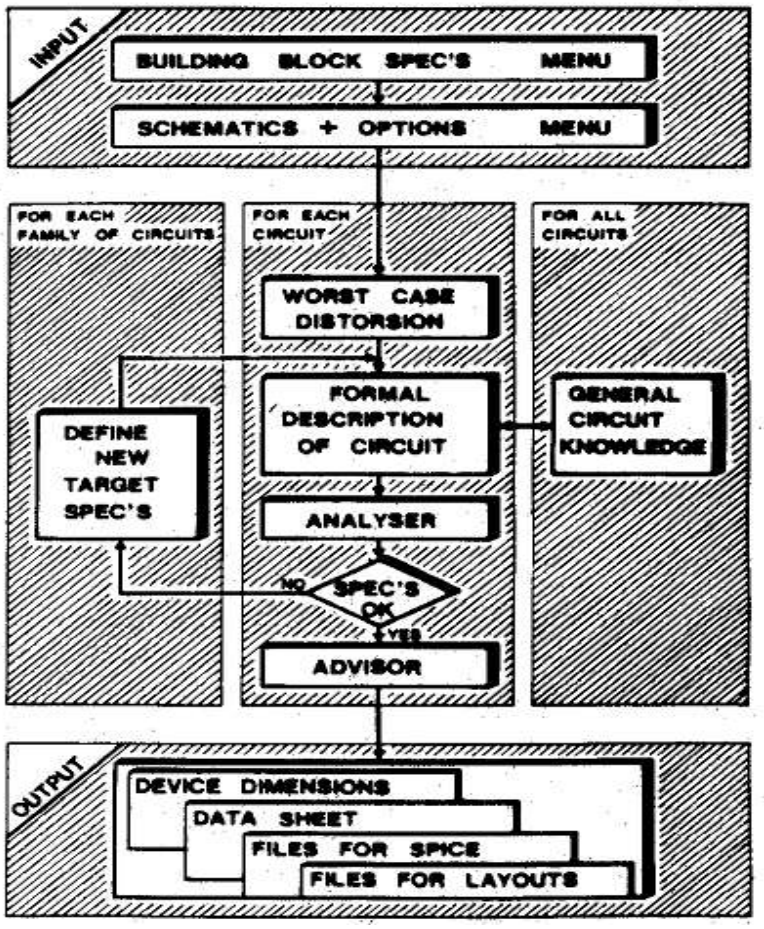


Figure 1.1 Overview of IDAC system chart

IDAC adopts a more knowledge based algorithm than an optimization one, by adopting equation based strategies and acquiring related circuit parameters e.g. minimum and

maximum value of the electrical parameters of MOS transistors, poly, well resistors, and layout rules, for computing circuit parasitics. In order to extract design worst-case solution, bias currents and mobility have been based on predictive equations which is not as accurate as models used nowadays in front-end simulators. These equations have been used to model the deterioration of chip performance under extreme high and low temperatures. IDAC system flow chart is shown in Fig. 1.1.

IDAC, KANSYS [4], and OPASYN [3] employed efficient equation based algorithms in terms of synthesis time and complexity; generating rough designs more quickly, creating an opportunity to explore design space. However as technology advances, it becomes much harder to render simple design equations to generate even rough specifications. OASYS[2] employed numerical optimization tools along with the circuit simulator to fine-tune device sizes in order to achieve the required performance.

OASYS [2], [28] adopted a hierarchical design strategy, in which analog circuit topologies are represented as a hierarchy of templates of abstract functional blocks. OASYS framework was based on three main ideas. Circuit topologies are selected from among fixed alternatives. A particular topology was chosen as a best candidate from which specifications were expected to be met. Secondly, the fixed alternatives for circuit topologies are identified hierarchically. A high level module was defined as an interconnection between sub-blocks. Finally, system level specifications could be then translated into sub-goals or specifications for the sub-block of a topology. The original motivation behind using separate selection and translation steps was to avoid the need to simultaneously design the interconnection and electrical characteristics of sub-blocks, where this hierarchical representation of topologies vastly simplifies the translation task since it tends to reduce the number of sub-blocks and simplify their connection. Hence, OASYS main contribution in the field of automated analog synthesis is the demonstration by which the analog behavior-to-structure synthesis problem could be recast in a highly structured form along with hierarchy as the key organizing principle. Translation involves knowledge of how performance specifications for a high-level block could be transformed into specifications for each sub-block, after which, these new specifications for each sub-

block would be used to design the transistors within each sub-block. The topology selection and translation process are shown in Fig. 1.2.

Each topology designed in OASYS has a design plan called plan steps in which three activities were performed. Heuristics, which are knowledge based decisions, make the design state more advanced by including some estimations that are based on the expertise of analog designers. After Heuristics planning, computation came next, where quantities like currents and biasing are computed from equations where sufficient information is available. These steps contributed mainly in assigning each sub-block certain specifications to achieve, and at last, a refinement step receives these new specifications, initiates sub-block design and retrieves the actual parameters that indicates the real performance of the circuit after synthesis. If simulated performance does not meet required specification, the topology is rejected and the search approach will be narrowed among the rest of the topologies.

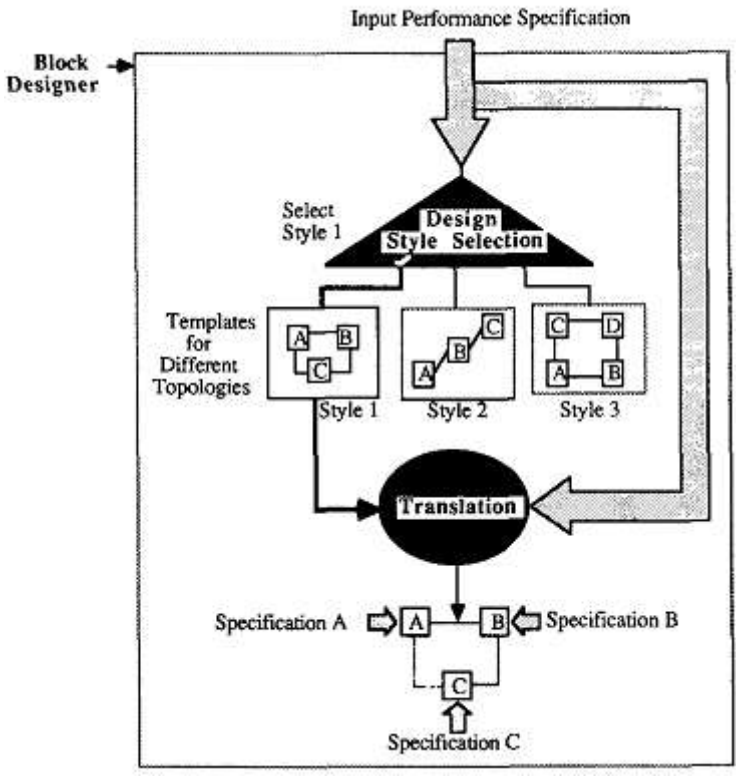


Figure 1.2. Topology selection and translation process in OASYS

In the selection phase, the algorithm can correct itself and return to a previous successful node in order to make an alternative topology style if one of the plan steps failed. On the basis of expert designers' observation in OASYS selection strategies, the tool complies with certain structural constraints such as; choosing between differential and single pair input nodes, which are totally user defined. Predicting performance limitations of circuits is defined as heuristic discrimination, which is based on expert designers' mature assessments of each topology. Obviously, it is the hardest type of discrimination since it is based on qualitative decisions which are hard to codify.

The last type of discrimination is in generate-and-test style which seems to be naive, but it is much more natural to compare crafted designs by hand to get an insight into which will work better. Basically, the major innovation behind OASYS [2] is the need to create an alternative to flat representations and to represent the tool in a more structured hierarchical form. However, Optimization of sub-blocks performances and employing knowledge on how choices made in one sub-circuit affects other sub-circuits is a hard problem.

KANSYS: Kanpur Analog Synthesis from the Indian Institute of Technology overcomes the drawbacks of hierarchical design by allowing the transfer of expertise among different sub-circuits translation algorithms empowering topologies translation in a more efficient way. In case of a failing specification in one sub-block, analytical equations are modified affecting all the sub-blocks. In addition, a search algorithm traversing the space in a hierarchy-aware fashion accounting for multi-objective optimization and process variations, is adopted in [28] using GP [29] and age-layered population structure [30]. However, quantifying circuit parameters dependency and higher order terms remains a hard problem. [21] proposed an approach to reduce independent variables and speed up design runtime by computing correction factor (S-factor) from transistor level simulations. By multiplying this factor by linearized circuit equations, accurate design can be achieved.

Other CAD tools adopted a design-to-layout approach [31-33] accounting for post-layout synthesis performance deterioration. AIDA [31] is the integration of GENOM-POF for circuit synthesis, and LAYGEN II for automated layout generation. GENOM-POF

performs circuit synthesis using multi-objective optimization approach, accounting for worst-case solution by exploring process, voltage and temperature variations in the design space. LAYGEN II generates a DRC proved layout based on the sized circuit descripted generate by GENOM-POF and high level layout guidelines. In circuit synthesis, the designer specifies design objective, number of optimization variables, the size of the design space, and the number of independent variables. Circuit parameters are optimized to obtain a set of Pareto-optimal solutions that fulfill all the constraints and shows different tradeoffs between circuit specifications. LAYGEN II uses the hierarchical template description, the sized devices, and the technology node kit to perform placement and routing followed by a validation step. The router uses placed modules, connectivity, symmetry, and sensitivity constraints in the optimization process. However, routing-aware-placement solution which ensures a better routability and reliability is not considered in the placement process.

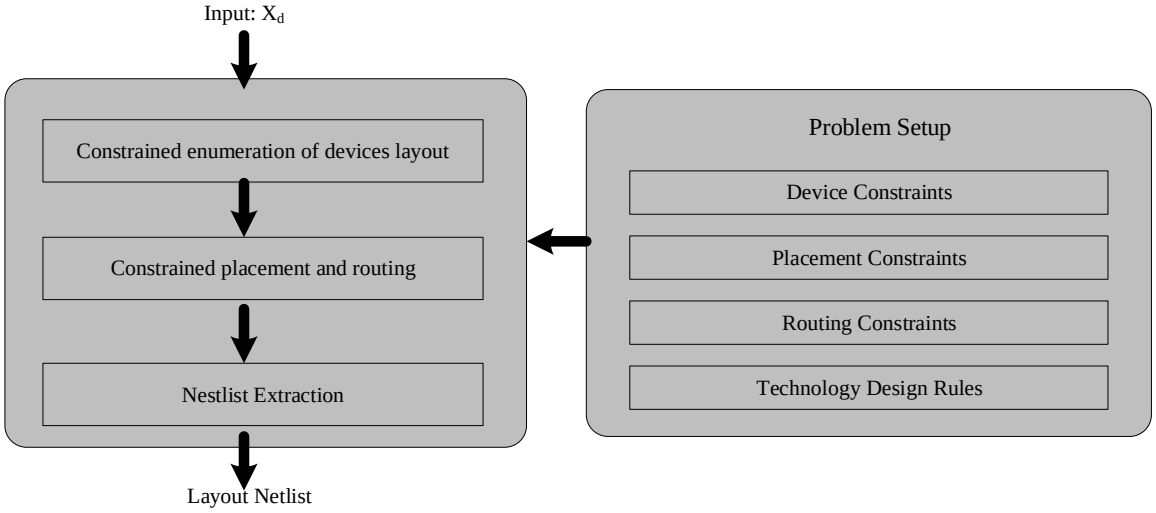


Figure 1.3. Layout optimization process

Dessouky *et al.* [32] proposed a layout-oriented circuit synthesis approach through passing the layout information at the beginning of the design phase. The approach guarantee a sized circuit performance that satisfied required specifications in the presence of layout parasistcs. Habal *et al.* [33] proposed an automated synthesis of circuit layout by investigating every feasible layout of each device, and the layout with best geometry are selected. The layout optimization process is driven by design, placement, and routing

constraints as shown in Fig. 1.3. Layout parasitics are extracted using an integral equation field solver without modeling. The first stage of circuit synthesis process involves formulating scalar minimization sub-problem on the basis of linearized objective function **f**, followed by solving the sub-problem using generalized boundary curve algorithm (GBC). Layout-driven circuit sizing flow chart is shown is Fig. 1.4.

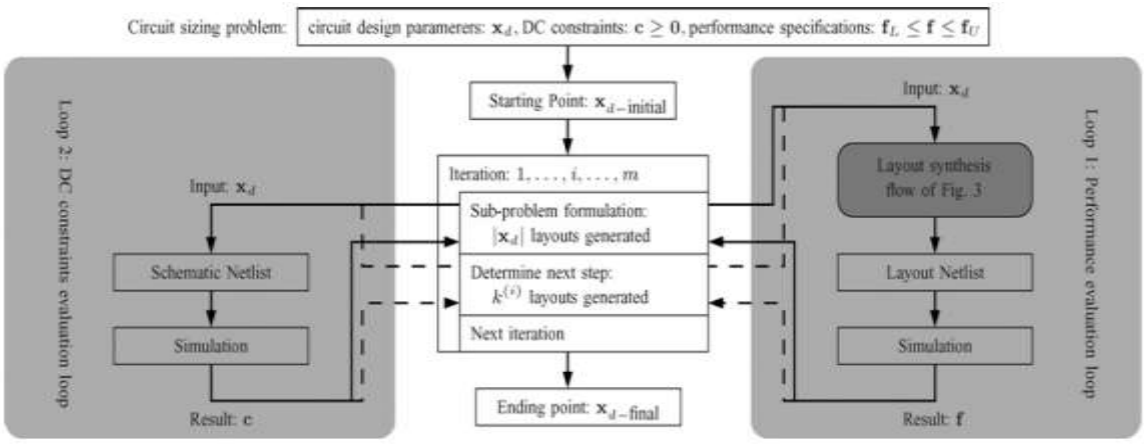


Figure 1.4. Layout-driven circuit sizing flow chart

A new layout is synthesized every iteration, where **f** has to be calculated for a new value of circuit parameters vector X_d by simulating generated layout netlist. Finite forward difference technique is implemented to calculate the gradient of performance **f** with respect to X_d. $k^{(i)}$ represents the design parameters vector at ith iteration at which performance is evaluated to determine next step by GBC.

Most of previous work in analog circuit synthesis have adopted hierarchical flow approach to optimize performance at cell level. Knowledge and expertise are required to be implemented in the tool for inter-processes optimization. However, even if applicable, generated sized circuits are outperformed by manual designs in terms of area and performance. Other tools adopted optimization algorithms e.g. simulated annealing, genetic algorithm which, if not implemented with enough design knowledge, may take a very long run time and may fail in achieving high performance. Numerical optimization can be adopted in circuit synthesis, since it always gives an output, i.e., if the specifications are not met, one has quantitative information of how far away the target is. It is easier

compared to other engines to introduce new specifications and schematics. However, such optimization is computationally extensive and hides different design tradeoffs between circuit parameters. Furthermore, the goal specification depends heavily on the initial solution. A fast and intelligent circuit synthesis remains a challenging problem despite the high quality of previous work.

Floorplanning and building block placement are becoming more crucial in physical design as the circuit sizes are growing rapidly and hierarchical design with IP blocks are widely used in to order to reduce design complexity. In VLSI design, floorplan and block placement are considered critical to the performance of design process. Classical floorplanning optimizes the area and wirelength of the chip blocks, and therefore, generates a compacted overlap-free placement of blocks. Floorplan representations are classified into two types; slicing and non-slicing representations. Slicing representation involves repetitively subdividing floorplan area horizontally and vertically into finite number of non-overlapping structures. Slicing brings faster packing runtime and higher convergence speed, compared to non-slicing representation. Number of blocks per slicing structure and, hence, cost evaluations are significantly reduced, where each structure is considered a separate solution space. However, optimal solution may not be achieved in the solution space of slicing structures. Slicing tree [9] and normalized polish expression [10] are popular slicing representation.

For considerably moderate solution spaces, Non-slicing floorplan can bring optimal solution, i.e. minimum area, interconnect delay, and minimum critical path, in a reasonable convergence time. SP [11], TCG [12], O-tree [13], and corner block list [14] are widely used non-slicing representations. Murata *et al.* [15] defined P-admissible solution space to distinguish non-slicing floorplans by the following four requirements;

1) Solution space is finite
2) Every solution is feasible
3) Packing and cost evaluation can be performed in polynomial time, and
4) Best evaluated packing in solution space corresponds to an optimal placement.

According to this classification, SP, TCG, O-tree, and BSG [16] are P-admissible while slicing tree, normalized polish expression, B*tree [17], and CBL are not. Since, slicing and

normalized polish expression do not generate optimal packing structures, they violate the conditions, and thus are not P-admissible representation.

Guo *et al.* [13] proposed an order tree (O-tree) representation for a left and bottom compacted placement with *n.logn* run time complexity. An admissible placement is a compacted one where blocks can neither move down nor left. According to the representation, each rectangular block is defined by its tuple $\{h_i, w_i\}$, where h_i and w_i are height and width of blocks respectively. A constraint graph of the placement is G=(V,E), where V presents each block in a form of a node. E represents geometric constraints between blocks which can be represented in a form of an edge drawn from the boundary of a block to another. Given an 8-node tree shown in Fig. 1.5, the placement can be encoded as (001010110100110,ABCDEFG). Starting from the root, node A is visited first and a bit '0' is recorded. Then node B is visited and a bit '0' is recorded. On the way back to the root, two bit '11' are recorded. The total number of possible configuration of an n-node tree is O($n!\, 2^{2n-2}/n^{1.5}$). Placement post packing may not be compacted, resulting in a mismatch between O-representation and its placement after a series of compaction operations. Similar to O-tree, B*tree solutions may not be feasible, and thus they are not P-admissible representations.

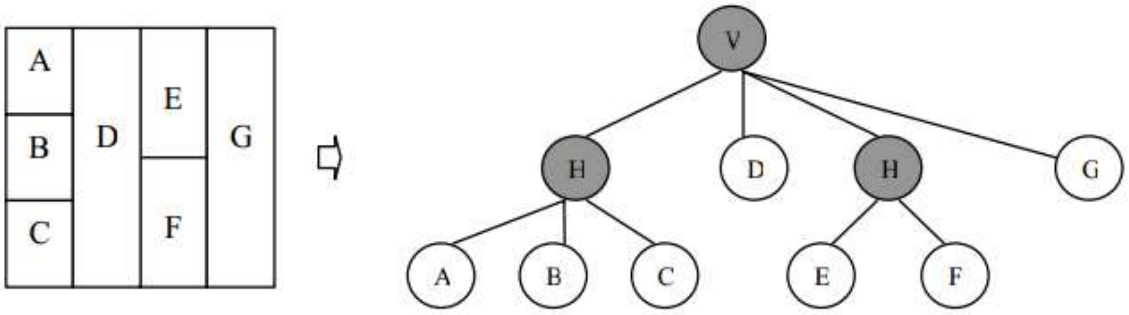


Figure 1.5. . Encoding of 8-node O-tree

Nakatake *et al.* [16] proposed a method of modules packing based on *bounded-sliceline grid* (BSG) structure. BSG is a meta-grid which does not contain physical dimensions, however, it is a topological grid composed of orthogonal lines called the BSG-units. BSG divides the planes into rooms associated with binary information coding the geometric relations between modules, such that any two rooms are uniquely in either relation.

Modules are assigned to BSG rooms in which they inherent the geometric relations between their rooms and other room in the meta-grid. Modules packing run time is O(n^2).

Hong *et al.* [14] proposed an efficient and effective topological representation of Non-Slicing Floorplan (CBL), which takes only linear time to derive modules placement from a representation. Unlike O-tree representation, corner block list defines the floorplan structure. Thus CBL is more flexible for floorplan optimization in terms of area and wirelength with different widths and heights of modules. Corner block list takes only $n(3 + [lg\ n])$ bits to describe, where $lg\ n$ is the minimum integral number which implies that corner block list need fewer bits to describe than SP and BSG needs. Corner block list performs recursive detection of corner block in a top-right mode to describe block placement. When the detection ends, block names and their orientations are concatenated in a reversed manner. The orientation of the corner block is defined by the joint of its left and right segment of the block and T-junction containing the joint. If the T-junction is rotated by 90 degress, the block is considered as vertically oriented, therefore its orientation is denoted by 0. The number of 1's in the T-junction list denotes the number of T-junctions attached to the block. Each string of 1's in T-junction list is ended by a 0 to separate it from other block detection. The advantage of CBL representation is that it does not only represent slicing structures, however, it can also represent non-slicing floorplan. The time complexity of floorplan realization is O(n) time which is better than SP, TCG, BSG, and TCG-S.

Lin *et al.* [12] proposed transitive close graph representation (TCG) for general non-slicing floorplans. TCG uses horizontal and vertical transitive closure graphs C_h, C_v to describe the geometric relation between modules of the placement. Lin *et al*. extended the concept of P-admissible representation to that of P*-admissible one by adding a fifth condition; both horizontal and vertical geometrical information between modules are defined in the representation. The fifth condition ease the handling of the floorplan design problems with further requirements such as module sizing and constraints, e.g., boundary, symmetry and proximity constraints. Thereby, the representation corresponds to the packing if the P*-admissible conditions are satisfied.

Consider the uncompact placement in Fig. 1.6. Since O-tree is not a P-admissible

representation, it is not flexible in handling uncompacted floorplan structure. Geometric relations between modules cannot be directly derived using O-tree and B*-tree representation unless the placement is packed. Whereas, TCG can handle P*-admissible representation due to its flexibility and elegant features. Some geometric features cannot be obtained by O-tree and B*-tree representations, implying that O-tree and B*-tree representations are harder to handle floorplan design and render better results in area and wirelength optimization problems. Furthermore, due to their compaction operation, perturbing the placement solution in O-tree and B*-tree may results in an unpacked solution implying that placement will not correspond to the representation after packing harming the solution structure and thus the optimum solution.

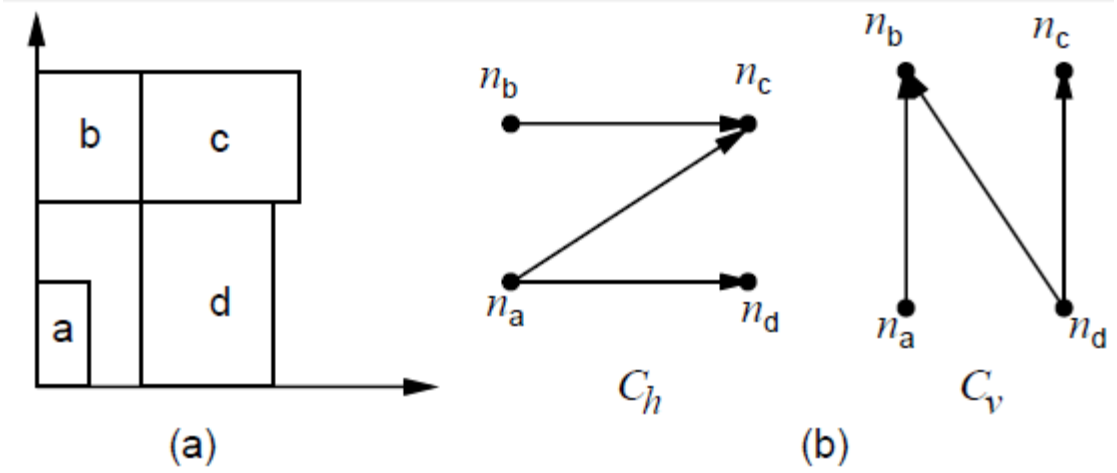


Figure 1.6. (a) Placement of four uncompact blocks. (b) The corresponding horizontal and vertical transitive closure graph C_h and C_v

TCG does not require any additional constraint graph for evaluation. Unlike SP, TCG supports incremental update after each solution perturbation and keeps positions of boundary modules as well as their geometric relation. Regarding SP, geometric relation among modules of a placement is not clear before packing and thus, SP constraint graphs are required to be generated from scratch for packing evaluation after each operation. CBL has a smaller feasible solution space and a faster packing scheme. However, CBL is not P-admissible as it represents general incompact placement. Given a TCG, its corresponding placement can be derived in O(n^2) by performing longest path algorithm, which is covered later in Chapter 4.

TCG representation is identified by three main properties; First, C_h and C_v are cyclic. A directed edge is constructed for each pair of nodes, which denotes modules in C_h and C_v graphs, according to geometric relations of these two modules. Since a pair of modules cannot be both below and above (left and right) to one another, the resulting C_h and C_v graphs must be acyclic. Second, for the aforementioned property 1, a pair of nodes must be connected by an edge in only one of the transitive closure graphs. Property 2 ensures that modules do not overlap since there is no horizontal and vertical relations between any pair of modules in a placement. The number of edges encoding the geometric relations between modules in a placement is $m(m-1)/2$, where m is the number of modules. Third, the transitive closure graph $C_h(C_v)$ is equal to itself.

TCG-S [18] a general floorplan representation was introduced, through integrating the properties of TCG and SP for a faster O($n\ log\ n$) runtime packing scheme using a balanced-binary search tree [19]. Same perturbing algorithm is adopted in both TCG and TCG-S representations, only the packing scheme in TCG-S is faster. Sequence Γ_- is the topological order of C_h and C_v closure graphs and therefore can be determined by C_h and C_v. Transparency of geometric relation between modules in placements and fast incremental update for cost realization are inherited from TCG. Furthermore, TCG-S shares the same feasibility properties with TCG. Given a floorplan, Γ_- can be derived by recursively extracting the module on the bottom-left corner of the placement as shown in Fig. 1.7. The run time of the extraction process is not indicated in [18]. C_h and C_v can be constructed based on Γ_- by constructing a directed edge from each node b_i before b_j in Γ_- in C_h (C_v) if $b_i \vdash b_j$ ($b_i \perp b_j$).

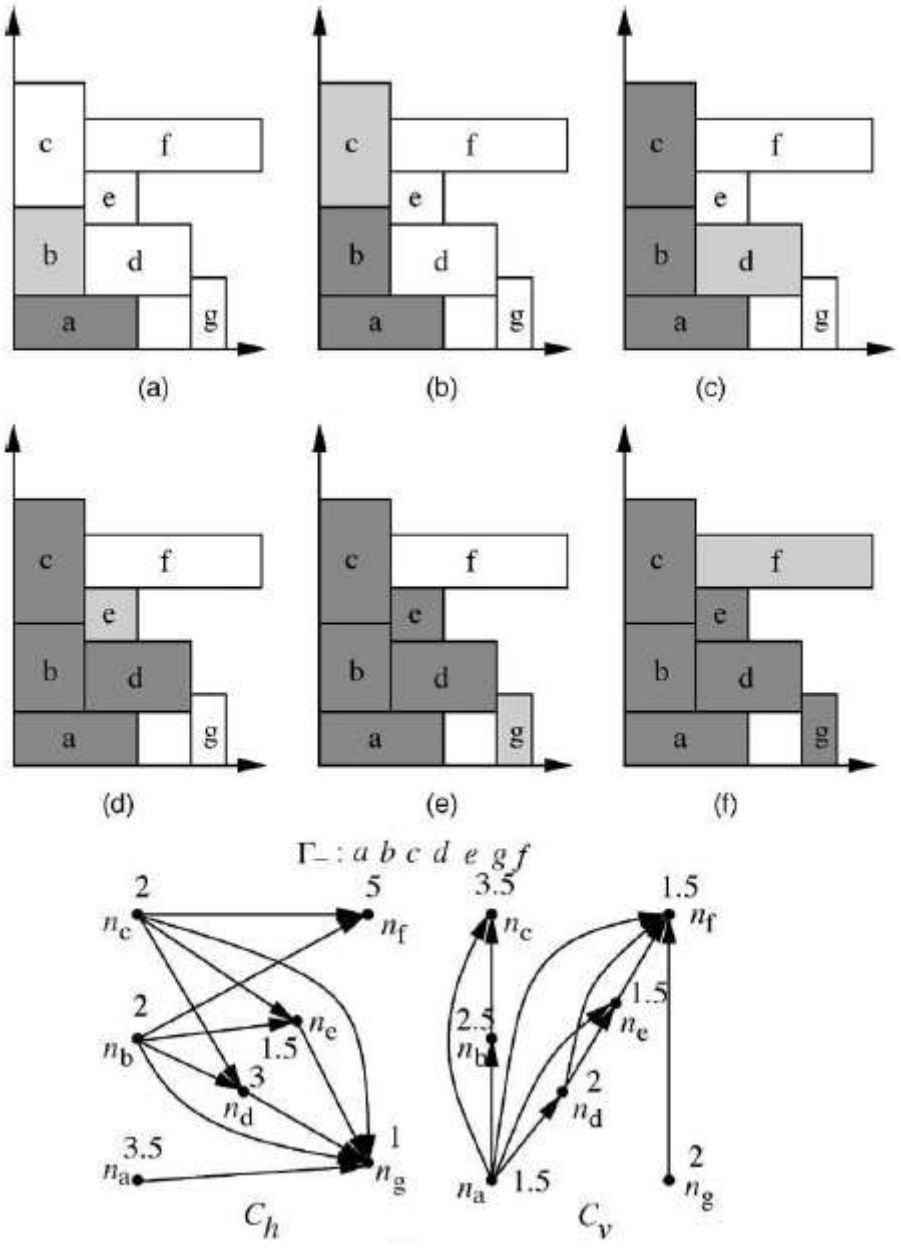


Figure 1.7. (a)–(f) Process to extract a Γ− from block placement. (g) Resulting TCG-S.

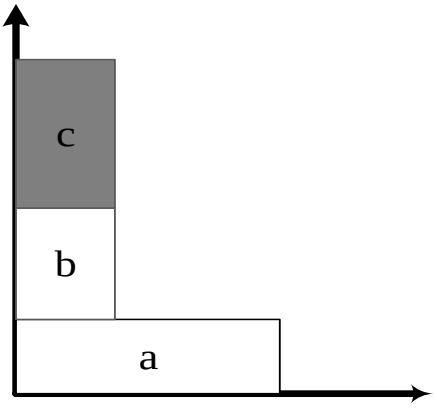


Figure 1.8. A block placement with sequence Γ_ ⟨a, b, c⟩

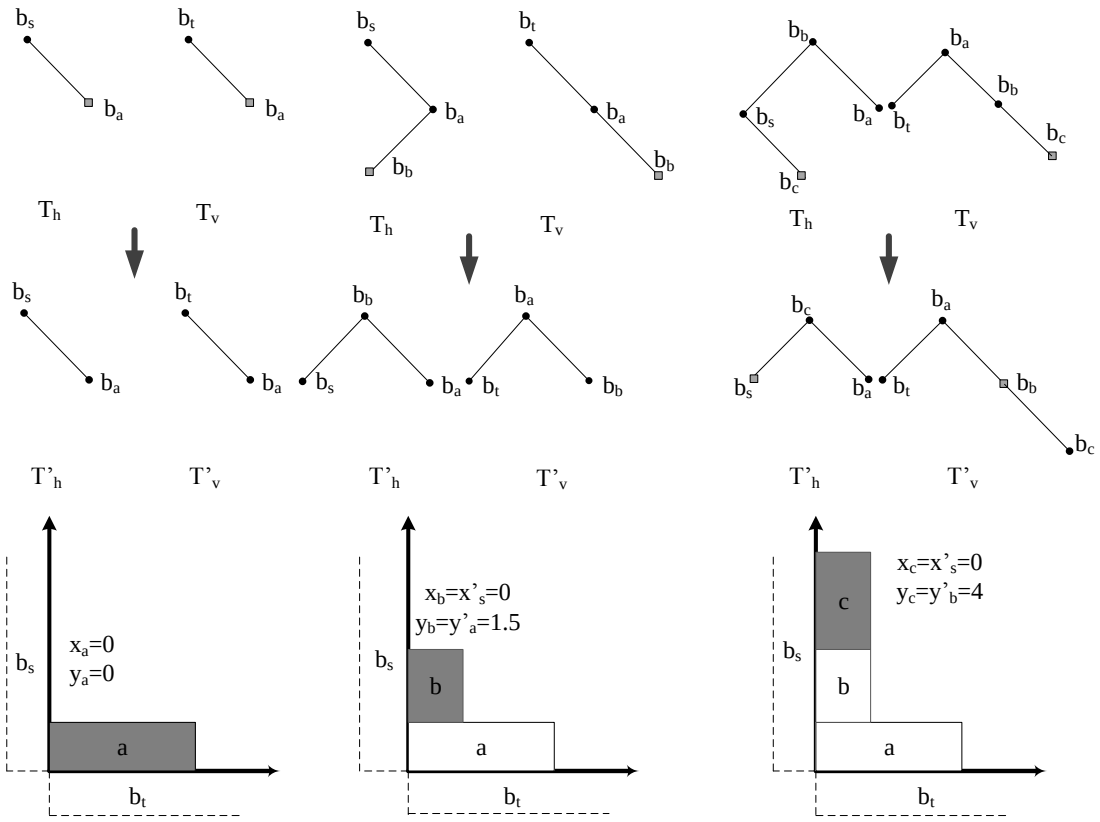


Figure 1.9. Packing scheme for the TCG-S of Fig. 1.8. In each step, the red-black trees T_h and , T_v corresponding to the R_h and R_v right after the module insertion, are shown. T'_h (T'_v) gives the resulting red-black tree after removing the modules no longer in R_h (R_v) and performing rotation to balance the tree. Note that, as a fundamental property of the binary search tree, the search-tree (in-order traversal) order is still maintained after the tree rotation.

Lin *et al.* [18] proposed an $O(nlogn)$ time packing scheme using Γ_- and horizontal and vertical contours R_h and R_v, where n is the number of modules in a placement. For each module in the sequence defined by Γ_-, the module is packed to a corner formed by two previously placed modules in R_h or R_v determined by the geometric relations defined by C_h or C_v.

Definition: Horizontal contour R_h and vertical contour R_v are lists of modules b_i's in which there does not exist any module b_j with $y_j \geq y_i'$, $y_j' \geq y_i'$ and $x_j \geq x_i'$, $x_j' \geq x_i'$ respectively.

The coordinates of the right and top boundaries modules in R_h and R_v are sorted and kept in a Red-Black search tree [19] T_h and T_v respectively. Module b_j is packed by searching for the last module b_p, where $b_p \vdash b_j$ or $b_p \perp b_j$, in order to compute the x-coordinate or y-coordinate of b_j according to the geometrical relation between modules b_p and b_j. Module b_k is traversed from its root to its right child if $b_k \vdash b_j$ ($b_k \perp b_j$), i.e. the right (top) boundary of module b_j is larger than that of module b_k. Therefore, b_j should be located in sub-tree of the search tree. The process alternates to the left child of b_k if $b_k \perp b_j$ ($b_k \vdash b_j$). Process continues until a leaf position is encountered and b_j is then considered the leaf node. Fig. 9 shows an example of TCG-S packing scheme of Fig. 8 with sequence Γ_- ⟨a b c⟩.

For placement, [34], [35] and [37] used a feasible sequence pair representation to develop symmetry constraint-driven placement tool. In order to illustrate a sequence pair representation which is symmetrically feasible, one would be tempted to perform minor changes to the search space exploration: if the current encoding proves to be consistent with the symmetry constraints then the cost of the placement configuration is evaluated and the annealing algorithm operates normally. Otherwise, the current encoding is infeasible (in symmetry point of view) and therefore, disregarded. Unfortunately, such a simple solution is not effective taking into account that the size of the search space without symmetry constraints is $O(n^2)$ (the total number of sequence-pairs). The size of the solution space becomes significantly smaller if the placement configuration must contain

a symmetry group. A better strategy is to explore only those sequence-pairs which comply with the symmetry constraints; recognize such sequence pairs and efficiently restrict the annealer exploration only to their subspace.

Whereas, [38] used the SP to tackle the placement problem with boundary constraints. A new constraint-driven placement approach is adopted in [36] based on constraints extraction via topology and signal flow analysis. Constraints are classified according to their critical levels and flexibility. The least flexible constraints has the highest priority in the optimization process.

In high performance circuits, it is required to places groups of devices symmetrically with respect to each other. The reason is to match the layout-induced parasitics and mechanical stresses in fabrication process within the symmetric groups. Failing to meet matching constraints lead to different values of parasitic resistances and capacitances at the differential output node. Such parasitic mismatch leads to higher offset voltage at the input differential pair and hence, lower gain and common mode rejection ratio. Balasa *et al.* [34] proposed a method to realize and handle symmetry constraints in block placement problem using sequence pair representation. Only the symmetry-feasible sequence pairs are explored, then passed to the annealer for area optimization.

Dong *et al.* [36] proposed a new constraint-driven placement technique for analog integrated circuits, where constraint are prioritized according to their critical levels. Such classification facilitates the search for better placement solution by reducing devices mismatch and critical paths parasitics indicated by the extracted constraints. Circuit constraints are extracted according to the topology and the signal flow analysis combined with heuristic knowledge of analog design. Symmetry and matching constraints are extracted using isomorphism graphs by primitive cell recognition in signal flow analysis. Constraints priorities are assigned values indicating their critical effect on performance of analog circuit, e.g., differential pair has a higher priority than other symmetry constraints. The objective function includes area, wirelength and critical path minimization using less flexible first algorithm (LFF).

Placement congestion problem is handled in previous literature [39-45] by employing routing-aware algorithms in the context of placement problem to guarantee the reliability and routability of the optimal placement solution. Constraints driven placement optimization are greedy and results in a compact placement solution, where its feasibility is questionable in terms of the reliability and the routability of the placed modules. In order to make the solution feasible, highly net-congested devices should be separated to create free spaces for successful routing. One approach [44] is to expand devices with high net congestion during placement and then release them to create routing channels. A probabilistic model is used in order to determine which devices need to expand and the corresponding expandable levels.

Operational amplifier is one of the most fundamental components in analog integrated circuit design. One of the essential tasks is to provide a high-performance opamp with high gain and bandwidth, and fast settling time. High-speed opamps use only one stage to reduce devices parasitics in order to achieve higher bandwidth. Telescopic opamps and folded cascode opamps are commonly used for this purpose.

The aim of this research is to present an optimized framework for operational amplifiers coupling both circuit design, accounting for process variation, and layout. Automated layout process includes floorplan design empowering area minimization, device placement accounting for symmetry constraints, and optimization-based transistor-level routing. Hence, assist in the introduction of the concept of optimized standard-cell, which is well-established in the digital flow, in the analog circuit design.

2.Circuit level synthesis

2.1. Folded Cascode OTA

2.1.1. Introduction

Folded cascode operational transconductance amplifier (OTA) is one of the most used topologies in analog circuits. It is a one stage amplifier since it has only one high impedance node at the output. It is considered as a self-compensated OTA due to the high output impedance. The compensation is usually achieved by the load capacitance, thus as the load capacitance becomes larger the operational amplifier becomes more stable but this comes at the expense of a lower bandwidth. Folded cascode OTA provides higher swing compared to the telescopic OTA as the input differential pair is in a separate branch making the output swing only limited by the overdrive voltage of four transistors instead of five, the case of telescopic OTA.

2.1.2. Basic Operation

The theory behind the folded cascode amplifier is to apply cascode transistors to the input differential pair and use complementary type of devices, converting applied input voltages to current and apply the result to a common gate stage. Fig. 2.1 shows the schematic of the folded cascode OTA. The static current consumption equation is given by:

$$I_{Tot.} = 2 * I_3 + I_{bias} + I_{CMFB} \quad (2.1)$$

The resistance at the output node can be calculated by:

$$R_{out} = R_{down} // R_{up} \approx \left(gm_5 ro_5 (ro_3 // ro_1)\right) // (gm_7 ro_7 ro_9) \quad (2.2)$$

Therefore, the DC gain can of the amplifier is given by:

$$A_v = Gm * Rout = -gm_1 \left(\left(gm_5 ro_5 (ro_3 // ro_1)\right) // (gm_7 ro_7 ro_9)\right) \quad (2.3)$$

Output swing which is difference between $Vout_{max}$ and $Vout_{min}$ is calculated as follows:

$$Vout_{max} = min\left(V_{b,1} + V_{th,7} \quad V_{dd} - V_{od,9} - V_{od,7}\right) \quad (2.4)$$

$$Vout_{min} = \max(V_{od,3} + V_{od,5} \quad V_{b,2} - V_{th,5}) \quad (2.5)$$

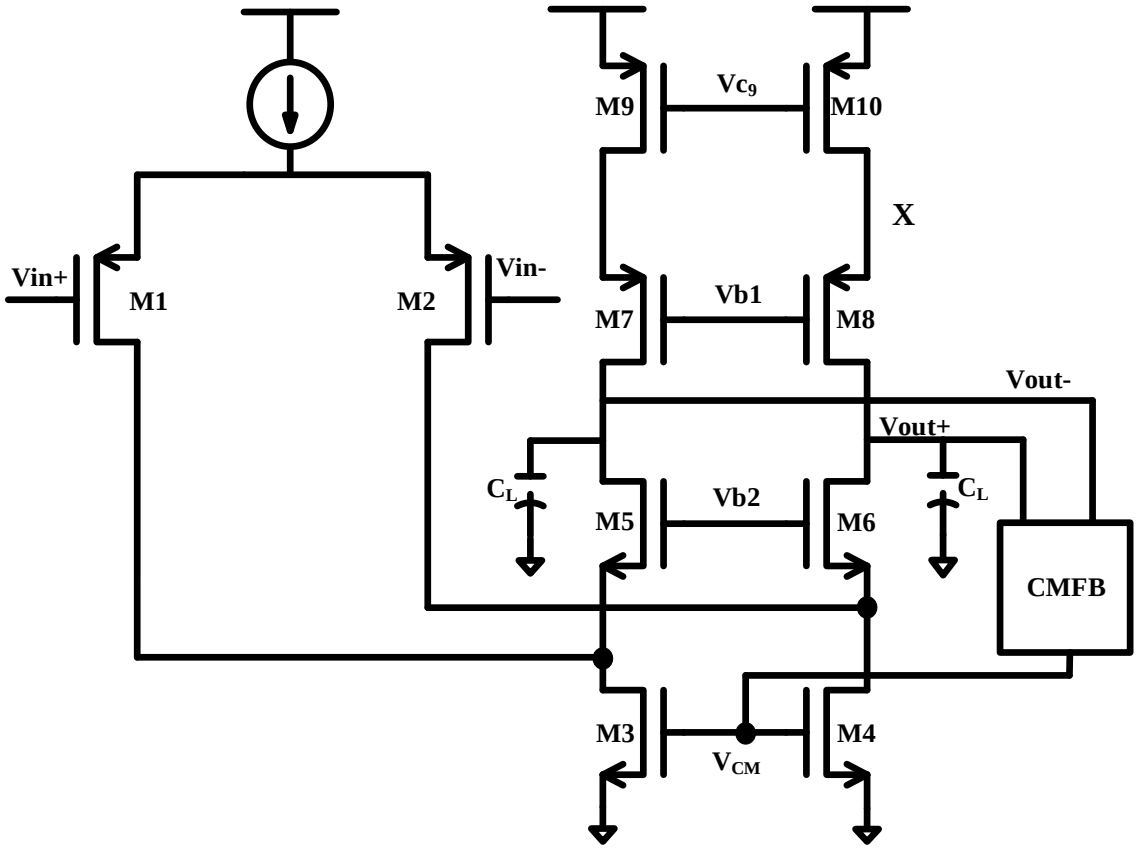


Figure 2.1. Folded cascode OTA circuit diagram

$Vout_{min}$ and $Vout_{max}$ are determined according to dc bias of the circuit, the maximum output voltage swing is achieved by the condition:

$$V_{b,2} \leq V_{od,3} + V_{od,5} + V_{th,5} \quad (2.6)$$

$$V_{b,1} \geq V_{dd} - V_{od,7} - V_{od,9} - V_{th,7} \quad (2.7)$$

Therefore, the absolute maximum output voltage swing is given by:

$$Swing_{out} = V_{dd} - (V_{od,3} + V_{od,5} + V_{od,7} + V_{od,9}) \quad (2.8)$$

Input common mode range which is the difference between $Vin_{CM,max}$ and $Vin_{CM,min}$ is calculated as follows:

$$Vin_{CM,min} = 0 \quad (2.9)$$

$$Vin_{CM,max} = V_{dd} - (V_{od,CS} + V_{gs,1}) \quad (2.10)$$

Since the input differential pair are PMOS, input common mode voltage level can be lowered to 0v without entering cut-off region of PMOS devices.

The maximum input common mode range is given by:

$$CMRange_{input} = Vin_{CM,max} - Vin_{CM,min} = V_{dd} - (V_{od,CS} + V_{gs,1}) \quad (2.11)$$

The unity gain frequency is calculated as follows:

$$f_u \approx \frac{gm_1}{2*\pi*C_{out}} \quad (2.12)$$

Bandwidth of the OTA, which represents its dominant pole, can be approximated by:

$$BW = f_{pd} = \frac{f_u}{A_v} \approx \frac{1}{2 * \pi * R_{out} * C_{out}} \quad (2.13)$$

Where,

$$C_{out} = C_L + C_{gd,5} + C_{gd,7} + C_{db,5} + C_{db,7} \quad (2.14)$$

The first non-dominant pole is calculated by:

$$f_{pnd,1} = \frac{1}{2 * \pi * R_{Fnode} * C_{Fnode}} \quad (2.15)$$

Where,

$$R_{Fnode} = (ro_1 // ro_3) // (\frac{1}{1 + (gm_5 + gmb_5)} * (1 + \frac{gm_7 ro_7 ro_9}{ro_5})) \approx \frac{1}{gm_5} \quad (2.16)$$

$$C_{Fnode} = C_{gs,5} + C_{gd,3} + C_{gd,1} + C_{db,3} + C_{db,1} \quad (2.17)$$

Therefore, $f_{pnd,1}$ can be approximated by:

$$f_{pnd,1} = \frac{gm_5}{2 * \pi * C_{Fnode}} \quad (2.18)$$

The second non-dominant pole at node X is calculated by:

$$f_{pnd,2} = \frac{1}{2 * \pi * R_X * C_X} \quad (2.19)$$

Where,

$$R_X = ro_9 // (\frac{1}{1 + (gm_7 + gmb_7)} * (1 + \frac{R_Y}{ro_7})) \quad (2.20)$$

$$R_Y = gm_5 ro_5 (ro_3 // ro_1) \quad (2.21)$$

$$C_X = C_{gs,7} + C_{gd,9} \quad (2.22)$$

Therefore, $f_{pnd,2}$ can be approximated by:

$$f_{pnd,2} = \frac{gm_7}{2\pi C_X} \quad (2.23)$$

The Phase margin, which determines the stability of the OTA, is given by:

$$PM = 180 - tan^{-1}\left(\frac{f_u}{f_{pd}}\right) - tan^{-1}\left(\frac{f_u}{f_{pnd,1}}\right) - tan^{-1}\left(\frac{f_u}{f_{pnd.2}}\right) \quad (2.24)$$

2.1.3. Common Mode Feedback

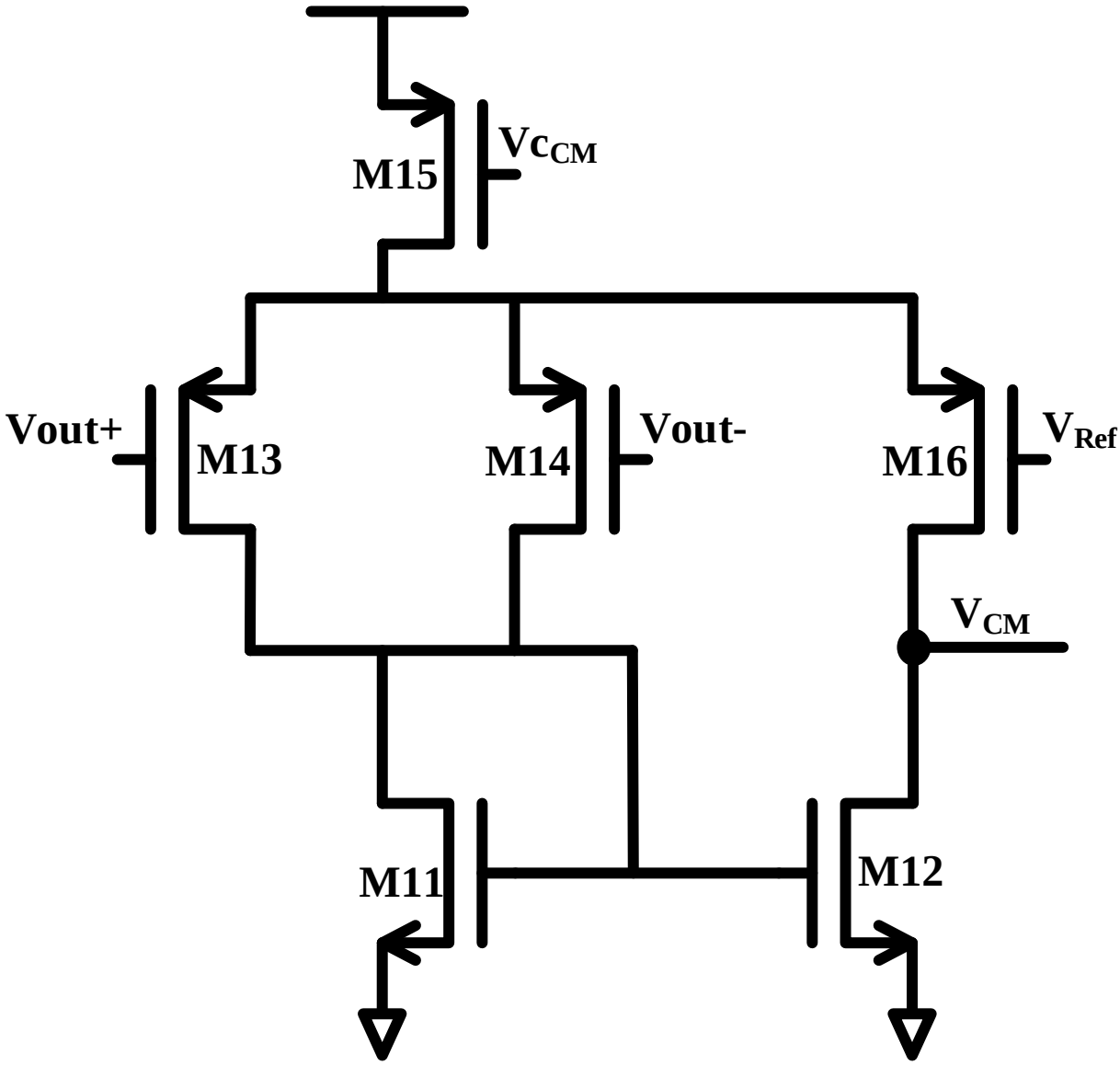


Figure 2.2. Common Mode FeedBack Circuit

The output voltage level of the amplifier is determined by the common mode level of the input differential signals. Since the output node is characterized by high impedance, it is hard to adjust the DC level of output. A negative feedback system is required to adjust the voltage at the output so that output current is the same at both sides of tail transistors. The output common mode of the amplifier is sensed by connecting them to a gate of sense transistors which are part of the CMFB circuits shown in Fig. 2.2.

2.1.4. Bias Circuit

Voltage biasing results in large current variations because of the process variations. Current biasing keeps the current constant in the device and independent of process variations. A simple current mirror has a low output impedance implying large variations in the mirrored current. A cascode current mirror is required to increase the output impedance and reduce the variations in DC output current, since the variations in the output voltage is reduced. Therefore, the current will be exactly mirrored the same to output transistor. Cascode current mirror circuit shown in Fig. 2.3 is used to simplify the design flow. The aspect ratio of the devices are chosen such that the sizing in both branches are related to the current by Eq. (2.26) and (2.27) . PMOS devices are required to mirror the current to the input current source device in input stage, cascode load, and CMFB circuit. Currents supplied by the bias circuit to the OTA are adjusted by sizing's ratio between the mirrored devices.

$$\frac{\left(W/L\right)_3}{\left(W/L\right)_0} = \frac{\left(W/L\right)_2}{\left(W/L\right)_1} \qquad (2.25)$$

$$\frac{I_2}{I_1} = \frac{\left(W/L\right)_2}{\left(W/L\right)_1} \qquad (2.26)$$

2.1.5. Advantages

- Large gain due to high output resistance.
- Moderate output swing.
- Can be used a unity gain buffer as output swing is relatively higher than other amplifier architectures, e.g. telescopic cascode.
- Higher bandwidth compared to telescopic cascode amplifier due to lower impedance at the output node.

2.1.6. Disadvantages

- Large power dissipation compared to telescopic and two stage miller compensated amplifiers.

- Lower phase margin compared to telescopic cascode amplifier due to higher capacitance value at folding node.

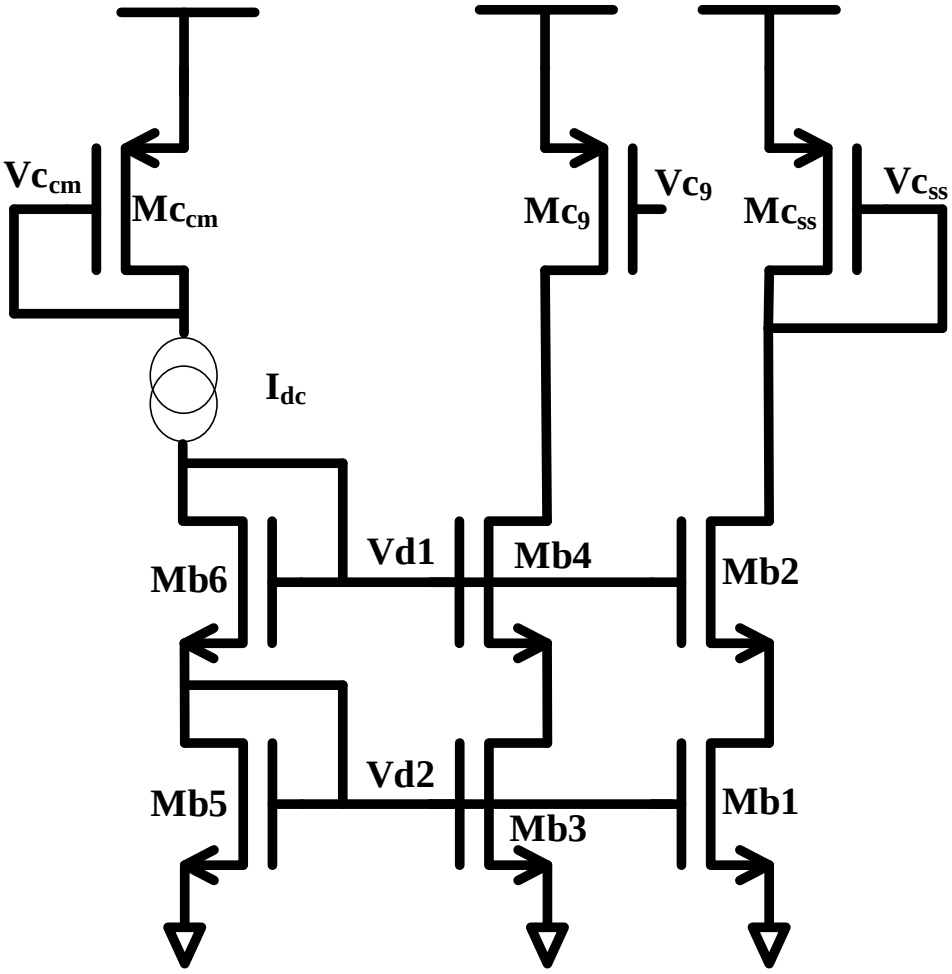


Figure 2.3. Folded Cascode OPAmp Bias circuit

2.2. Sensitivity Analysis

The high number of parameters in analog circuit complex models constitute a significant problem in their design, since the parameter estimation becomes a high dimensional, multi-modal and predominantly a non-linear problem. Approaches are adopted to resolve the problem by implementing a wide range of optimization algorithms, which are neither feasible nor efficient in determining the performance dominating parameters in the non-monotonically, multi-dimension design space. A sensitivity analysis facilitates the search for the most influential parameters in the circuit, allowing the reduction of total number of parameter in the optimization process, or quantify some interactions effects between input parameter within the circuit model. Sensitivity analysis (SA) tools are of immeasurable value, allowing the study of how the uncertainty in model output can be apportioned to difference source of uncertainties represented in the model inputs. SA has a wide scope of usage and applications; model understanding, verification, simplifying models and prioritization of model parameters.

Definition of sensitivity analysis involves models, inputs and outputs. In order to define model input with respect to uncertainty and the sensitivity analysis, a model can be classified into:

- Diagnostic or prognostic: in which the model can be used to understand a law or in predicting the behavior of the system given an understandable law. Models thus can range from speculations to accurately predicting a system.
- Data-driven or law-driven: A law-driven model puts together trusted laws which have been attributed to the system, in order to predict its behavior. A data-driven model treats the components of a system as a signal and derives its properties statistically. Law-driven models have the higher capacity to understand system behavior under unobserved circumstances. Whereas, data-driven models is only limited to the behavior associated with data in their estimations.

Definition of model input depends on the model under study. In order to have an acceptable grasp of the uncertainty principle and sensitivity analysis, model input is defined as any parameters that derives variations in the model output.

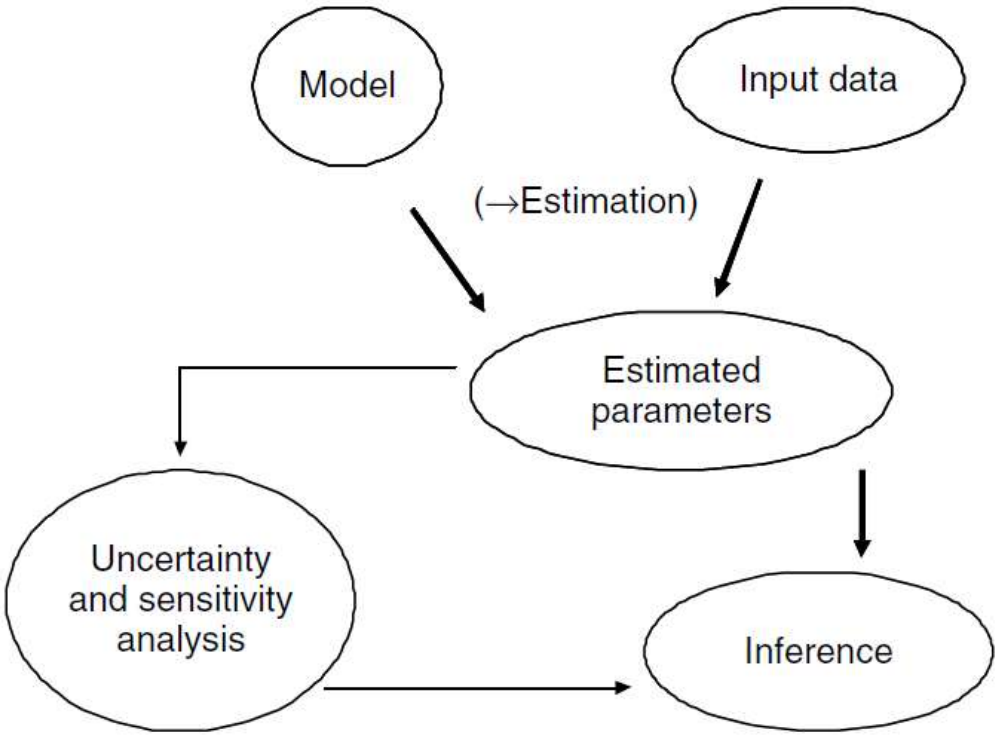


Figure 2.4 Parameric bootstrap version of uncertainty and sensitivity analysis

Consider the flow chart in Fig. 2.4. At the end of the estimation step, parameter values as well as their error are known. The model is considered true and uncertainty analysis is performed through propagating uncertainty in the parameters of the model, all the way to the model output. From uncertainty analysis, the average output and standard deviation is computed. This analysis can be repeated with sufficiently large number of parameters variations, hence it is called 'parametric bootstrap'. It is a process of repeatedly propagating the uncertainty in the parameters through the model, each iteration computing the average output and the standard deviation, in order to increase the accuracy of the output values and hence reduce errors. Sensitivity analysis is then performed to determine which of the input parameters are more important in influencing the uncertainty of the model output. It

is of high significance that objectives and input parameters for uncertainty and sensitivity analysis are carefully selected. The more parameters considered as input, the greater and the more accurate a variance to be expected in the model output.

2.2.1. Classification of sensitivity analysis

Sensitivity analysis can serve a number of useful purposes in modelling. It can uncover errors in the model, establish priorities for research, and simplify models. SA can be categorized into two approaches; local and global analysis. Local analysis studies the small input perturbations on the model output which occur around nominal values, e.g., the mean of an input variable. Local SA is considered a deterministic approach, where output variations due these small perturbation are obtained by computing the partial derivative of the model at a certain point. Derivative-based approach has the advantage of being efficient in terms of run time. The model needs to be executed few times according to the dimension of the array of derivatives. However, the failing part of this approach is that it is unaware if the model input is uncertain or if the model is of unknown linearity. Derivatives are only informative around the nominal value where they are computed and hence, do not provide for any exploration of the rest of the space of the input parameters. Such disadvantage has a minor effect or even no effect for linear systems, however, it matters greatly knowing that the system is non-linear and non-monotonic.

The very basic definition of sensitivity Index (SI) is given by:

$$SI_i = \frac{y_i^{max} - y_i^{min}}{y_i^{max}} \qquad (2.27)$$

Where y_i^{max} is the maximum between y(x_i^{min}) and y(x_i^{max}), and y(x_i) is computed at nominal value x_0. Variable x_i is moved one-at-a-time (OAT) to its respective x_i^{max} and x_i^{min}.

2.2.2. Local Sensitivity analysis

According to local sensitivity analysis, a simple calculation of sensitivity of f(x) can be given considering second order Taylor series, is given by:

$$f(x_0 + \Delta) = f(x_0) + \sum_{i=1}^{k} \frac{\partial f(x_0)}{\partial x_i} \Delta_i + \frac{1}{2} \sum_{i=1}^{k} \sum_{j=1}^{k} \frac{\partial^2 f(x_0)}{\partial x_i \partial x_j} \Delta_i \Delta_j \quad (2.28)$$

Using the OAT approach realizing k+1 runs, a finite difference approximation to the first order local sensitivity can be computed as follows:

$$\frac{\partial f}{\partial x_i} \cong \frac{y(x_{0,i} + \Delta_i) - y(x_{0,i})}{\Delta_i} \quad (2.29)$$

For uncorrelated inputs variables, expectation vector and the variance of the function f(x) is defined as:

$$E(Y) = f(x_0) \quad (2.30)$$

and

$$var(Y) = \sum_{i=1}^{k} \left[\frac{\partial f(x_0)}{\partial x_i} \right]^2 . var(x_i) \quad (2.31)$$

In order to overcome the large limitation of local SA, which only considers local variations accompanied with limited range linearity calculations, global SA has been introduced in a statistical framework. Global SA considers the whole range of variations of input variables, therefore, can be used in the study of models in order to identify and prioritize the most influential inputs parameters, identify non-influential parameters which has a very minor effect on the output uncertainty in order to be fixed during design space exploration. Global SA can also be used to map the output behavior in function of input variables by focusing on certain range of inputs values, and the calibration and validation of model equations. The aim of this section is to provide a review on global sensitivity analysis which is one of the techniques in ANOVA family.

2.2.3. Global Sensitivity analysis

2.2.3.1. Regression-based correlation analysis

The correlation coefficient designate the strength and direction of a linear relationship between two random variables. The best known coefficient is the Pearson product-moment correlation coefficient, which calculated by dividing the covariance of the two variables by the product of their standard deviations. Pearson correlation coefficient is defined as:

$$\rho_{X,Y} = \frac{E(X,Y) - E(X)E(Y)}{\sqrt{E(X^2) - E^2(X)}\sqrt{E(Y^2) - E^2(Y)}} \quad (2.32)$$

Combining MonteCarlo simulation, Person correlation coefficient is given by:

$$r_{x,y} = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2 \sum_{i=1}^{N}(y_i - \bar{y})^2}} \quad (2.33)$$

Where $\bar{x}$ is the mean value of x_i and $\bar{y}$ is the mean value of y_i. Correlation coefficients values range in the interval [-1,1], where 0 indicates a linear relationship and (-1,1) indicate a strong relationship between random variables under study. Consider a variable Y dependent upon number of variables $X = (X_1, X_2, X_3, \dots, X_n)$, hence the correlation coefficient can be used as a sensitivity measure.

$$S_i = \rho_{X,Y} \quad (2.34)$$

The correlation is powerful measure to summarize linear relationships between variables. However, in case of non-linearity it may lead to wrong conclusions. Hence, a correlation analysis cannot replace individual examination of data.

Pearson correlation coefficient is combined with regression coefficient obtained by linear regression analysis. Regression analysis indicates the strength and direction of a relationship between two random variables X and Y as well. Random variable is defined to to be dependent and modeled as a function of an independent variable, its parameters, and a random error term. In linear regression, in order to model n date points there is one independent variable x_i, two parameters a and b and an error term ε_i.

$$y_i = a + bx_i + \varepsilon_i \quad (2.35)$$

In order to compute a and b, least square method is used as follows:

$$\hat{a} = \bar{y} - \hat{b}\bar{x} \quad (2.36)$$

$$\hat{b} = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{N}(x_i - \bar{x})^2} \quad (2.37)$$

The interrelation between linear regression and Pearson correlation coefficient is defined by

$$\hat{b} = r_{x,y}\frac{S_y}{S_x} \quad (2.38)$$

Where S_y and S_x are the standard deviation of the n data points.

The proportion of variability in the data processed by the linear regression is defined by the coefficient of determination $R_{x,y}^2$. The variability of date is measured by computing the residuals as follows:

$$\hat{u}_i = y_i - (\hat{a} + \hat{b}x_i) \quad (2.39)$$

Hence, coefficient of determination $R_{x,y}^2$ can be calculated as follows:

$$R_{x,y}^2 = 1 - \frac{\sum_{i=1}^{N}\hat{u}^2{}_i}{\sum_{i=1}^{N}(y_i - \hat{y})^2} \quad (2.40)$$

Where $R_{x,y}^2$ is the square of the Pearson correlation coefficient $r_{x,y}$, in case of linear regression.

2.2.3.2. Variance-based approaches

The models under study are described by a function $Y = f(X)$, where $X = (X_1, X_2, X_3, \dots, X_n)$ and X is a random input vector consisting of n random variables. $Y = (Y_1, Y_2, Y_3, \dots, Y_m)$ denotes the random output vector functions of random variables. $f(X)$ can be decomposed into summands of increasing order components.

$$f(X) = f_0 + \sum_{i=1}^{n} f_1(X_1) + \sum_{1 \le i \le j \le n} f_{i,j}(X_i, X_j) + \cdots + f_{1,2,\ldots,n}(X_1, \ldots X_n) \qquad (2.41)$$

Each random model response Y_j, where j = 1,2,…m, can be characterized by its variance V^j. Each variance V^j is decomposed into partial variances corresponding to the single random input variables $X_1, X_2, X_3, \ldots, X_n$ according to equation (2.42), and to relate each partial variance to a single sensitivity measure according to equation (2.43):

$$V^j = \sum_{i=1}^{n} {V_i}^j + \sum_{1 \le i \le k \le n} {V_{i,k}}^j + \cdots + {V_{1,2,\ldots n}}^j \qquad (2.42)$$

$$S_{i_1,\ldots,i_s} = \frac{{V_{i_1,\ldots,i_s}}^j}{V^j} \; where \; 1 < i_1 < i_2 < i_3 \ldots < i_s \le n \qquad (2.43)$$

Each of the sensitivity measures calculated by equation (11) describes which amount of each variance V^j is generated due to the randomness of the associated random input variables and their mapping onto the output variables. As special case the sensitivity measures S_i^j describing the sole influence of the single input variables X_i are called the main effects. Whereas, sensitivity measures $S_{i_1,\ldots,i_s}$ describing the influence of combinations of input variables are denoted as interaction effects.

All partial sensitivity indices S_i^j are summed up to the total sensitivity measure S_{Ti}^j in order to evaluate the total effect of the single input variable X_i. Hence, the total sensitivity measures consider the interactions among input variables. In order to quantify which amount of each variance V^j is generated due to a single input variable X_i, the corresponding total sensitivity measure S_{Ti}^j can be normalized as follows:

$$norm\left(S_{Ti}^j\right) = \frac{S_{Ti}^j}{\sum_{k=1}^{n} S_{Tk}^j} \qquad (2.44)$$

2.3. Overview of OASYN framework

Figure 2.5 shows an overview of the OASYN framework. The tool acquires a topology from two well-known operational amplifiers structures; the Folded Cascode and the Two Stage Miller compensated amplifiers, according to designer preferences, along with required specification, e.g. gain, GBW, phase margin, output swing, slew rate, load capacitor, technology node, input common mode voltage level, and maximum power consumption. The tool acquires connectivity electrical constraints, e.g. max current density information in each circuit net, and matching constraints for device group placement along with matching styles. Circuit synthesizer generates a rough initial estimate sizing based on the analytical circuit equations. Then, the tool undergoes sensitivity analysis employing Sobol indices in the circuit sizing optimization engine, and a Pareto-optimal set is generated for immediate translation of specs to fully sized topology. To the authors' knowledge, this is the first work that examines the whole design space through sensitivity analysis in order to account for uncertainty of the non-linear behavior of analog circuits, by quantifying higher order interactions between parameters of the circuit taking into consideration extreme eprocess, supply, and temperature variations.

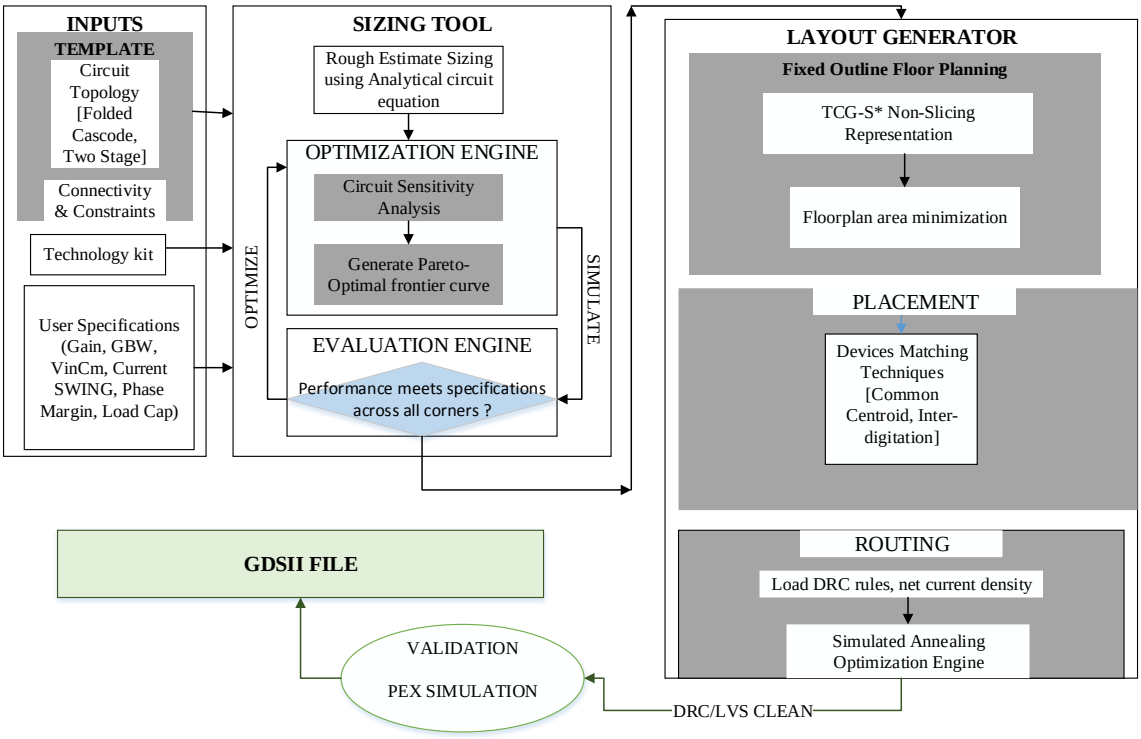


Figure 2.5. Overview of the OASYN framework

Layout generator tool consists of three main processes. First, a fixed-outline floorplanner employing multi-objective optimization on area and wirelength, accounting for block placement matching constraints, is implemented. This paper proposes a new, simple, efficient, and fast floorplan solution perturbing algorithm with O(n) runtime complexity, for fast realization of incremental update for cost evaluation, called TCG-S*. The algorithm integrates the advantages of TCG and SP representations, and eliminates their disadvantages, into a superior topology update scheme which facilitates the search for optimal desired floorplan.

In order to enhance routability and reliability of the packed optimal placement solution, a routing-aware algorithm is implemented within the placement process contemplating the congestion problem, smoothing the densities between placed blocks and preserving the relative location of the modules. An annealing-based detailed net routing is then executed to generate a free DRC layout.

2.4. Circuit sizing tool

The main purpose of the sensitivity analysis is to determine the most influential model parameters affecting a model response. Hence, reduce the computational complexity in optimization. Local and global analysis are major constituents of sensitivity analysis. The high priority parameters in one part of design space may not be the same in another, highlighting the importance of global SA. In addition, importance of a subset of variables may be subject to the interactions between these variables rather than the sum of the individual variables importance. Sensitivity analysis based optimization is employed in previous works [5], [7], [8], [46], and [47]. Variance-based Sobol method efficiently quantifies synergic effects along with uncertainties in the model input and their effect on the model output.

2.4.1. The Sobol' Sensitivity Analysis

The Sobol' decomposition [51, 52] is one of the family of ANOVA techniques. The Interaction of two or more parameters are denoted as Sobol' indices. The function F(ξ) of a set of input variables ξ_i, where Ω_d is a dimensional range and d is the total number of input variables, is defined by

$$F(\xi) = \sum_{u \subseteq (1.2,\dots d)} F_u(\xi_u) \qquad (2.45)$$

Where u is a set of integers, $\xi_u = (\xi_{u_1}, \dots, \xi_{u_s})$ and s = $|u|$. In order to calculate the effect of certain input variables on the output uncertainty, u represents these sets of variables as a subset of the whole variables set, presented in Eq. (2.45), as will be shown later in the section. Eq. (2.45) is decomposed as follows:

$$F(\xi_u) = F_0 + \sum_{1 \le i \le d} F_{u_i}(\xi_{u_i}) + \sum_{1 \le i \le j \le d} F_{u_{ij}}\left(\xi_{u_i}, \xi_{u_j}\right) + .. \\ + F_{u_{12\dots d}}(\xi_{u_1}, \dots, \xi_{u_d}) \qquad (2.46)$$

In this expansion, the individual terms can be calculated according to

$$F_0 = \int_{\Omega^d} F(\xi) d\xi \quad (2.47)$$

$$F_u(\xi_u) = \int_{\Omega^{d-\}u\}} F(\xi_u) d\xi_{\sim u} - \sum_{\substack{v \subset u \\ v \neq u}} F_v(\xi_v) \quad (2.48)$$

Where $\xi_{\sim u}$ is ξ with set u excluded

$$\xi_{\sim(b)} = (\xi_1, \dots, \xi_{b-1}, \xi_{b+1}, \dots, \xi_d) \quad (2.49)$$

Equation (2.50) defines the total variance of the output function F(ξ), denoted by D. D_u denotes the partial output variance in response to a set of input variables.

$$D = \int_{\Omega^d} F^2(\xi) d\xi - F_0^2 \quad (2.50)$$

$$D_u = \int_{\Omega^{|u|}} F_u^2(\xi_u) d\xi_u \quad (2.51)$$

D_u can be represented as recursive function of conditional variances:

$$D_u = V(E[y|\xi_u]) - \sum_{\substack{v \subset u \\ v \neq u \\ v \neq 0}} D_v \quad (2.52)$$

And therefore, D can be represented as the summation of the variances D_u:

$$D = \sum_{\substack{u \subseteq \{1,2,\dots d\} \\ u \neq 0}} D_u \quad (2.53)$$

D_u measures the variance of output $F(\xi)$ according to the interaction between elements of u, subtracting the individual effect of elements $v \subset u$. The Sobol' sensitivity indices can be calculated by:

$$S_u = \frac{D_u}{D} \quad (2.54)$$

$$\sum_{\substack{u\subseteq\{1,2,\ldots,d\}\\ u\neq 0}} S_u = 1 \qquad (2.55)$$

Where S_u measures the sensitivity of $F(\xi)$ by the interaction of elements of u, excluding the effect each variable separately have on output function variance. There are $2^d - 1$ sensitivity indices required to be calculated in order to determine the most significant design parameters.

2.4.2. Computation of Sobol' Indices by Monte-Carlo Sampling

Calculation of the variances using integrals is extensive process since circuit model equations are complex and non-linear. Therefore, a sample set of n realizations of input variables ξ_u is considered to calculate the average E[y] and the variance D.

$$D = E[y^2] - E[y]^2 \qquad (2.56)$$

According to Eq. (2.47) and (2.56), the sampled estimates of F_0 and D are:

$$\hat{F}_0 = \frac{1}{n}\sum_{i=1}^{n} F(\xi^{(i)}) \qquad (2.57)$$

$$\hat{D} = \frac{1}{n}\sum_{i=1}^{n} F^2\left(\xi^{(i)}\right) - \hat{F}_0{}^2 \qquad (2.58)$$

According to Eq. (2.52), Estimate of D_u can be calculated by finding an expression for the conditional variance estimate as follows:

$$V(E[y|\xi_u]) = E[E[y|\xi_u]^2] - E\left[E[y|\xi_u]\right]^2 = E[E[y|\xi_u]^2] - E[y]^2$$

$$\approx \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{n}\sum_{j=1}^{n} F\left(\xi_{\sim u}^{(j)}, \xi_u^{(i)}\right)\right)^2 - F_0^2 \qquad (2.59)$$

However, time complexity of computing conditional variances is $O(n^2)$. Sobol [51] proposed a faster method to calculate the variances using Monte-Carlo sampling technique using two sample sets $\xi^{(i)}|_{i=1}^{n}$ and $\eta^{(i)}|_{i=1}^{n}$.

$$E[E[y|\xi_u]^2] = E\big[E[y|\xi_u]\,E[y|\xi_u]\big] = \int \left(\int F(\xi_{\sim u}, \xi_u) d\xi_{\sim u}\right) * \left(\int F(\xi_{\sim u}, \xi_u) d\xi_{-u}\right)$$

$$= \int\int\int F(\xi) F(\eta_{\sim u}, \xi_u)\, d\xi d\eta_{\sim u} \qquad (2.60)$$

Substituting Eq. (16) in Eq. (8), estimate of D_u becomes:

$$\widehat{D}_u = \frac{1}{n}\sum_{i=1} F\left(\xi^{(i)}\right) F\left(\xi_u^{(i)}\right) - \sum_{\substack{v \subset u \\ v \neq u}} \widehat{D}_v \qquad (2.61)$$

Where

$$(\xi_b)_u^{(i)} = \begin{cases} \xi_b^{(i)} & b \in u \\ \eta_j^{(i)} & otherwise \end{cases} \qquad (2.62)$$

Therefore,

$$\widehat{D}_{\{b\}} = \frac{1}{n}\sum_{i=1} F\left(\xi_1^{(i)}, \dots, \xi_d^{(i)}\right) * F\left(\eta_1^{(i)}, \dots, \eta_{b-1}^{(i)}, \xi_b^{(i)}, \eta_{b+1}^{(i)}, \dots, \eta_d^{(i)}\right) - \hat{F}_0{}^2 \qquad (2.63)$$

2.4.3. Circuit Sizing Algorithm

Algorithm 2.1: Monte_Carlo(U ξ_Sample η_Sample)

1. // Initialize Sum with 0

2. FOR i 0 n-1 DO // no. of samples

3. FOR j 0 Var_NUM DO // total number of variables

4. SamList1 = concat{SamList1 ξ_Sample[i,j]};

5. if (j on U) THEN

6. SamList2 = concat{SamList2 ξ_Sample[i,j]};

7. ELSE SamList2 = concat{SamList2 η_Sample[i,j]};

8. Out_Sim1 = SIMULATE(SamList1);

9. Out_Sim2 = SIMULATE(SamList2);

10. Sum = Sum+Out_Sim1*Out_Sim2;

11. RETURN Sum/n;

Algorithm 2.2: Sobol_Decomp(List partial rest Result)

1. //Initialize partial , res, Result with nil

2. FOR i 0 length(List)-1 DO

3. n = nth(i List);

4. rest = List;

5. FOR j 0 i DO

6. rest = REMOVE(nth(j List) rest); // delete jth element

7. Result = Sobol_Decomp(rest 0 concat{partial n} nil concat{Result concat{partial n}});

8. RETURN Result;

Algorithm 2.3: Sobol_Var(list(U))

1. // calculate Variance D_U

2. FOR i 0 length(U)-1 DO

3. MC = Monte-Carlo(nth(i U) ξ_Sample η_Sample);

4. if (length(i U) ==1 THEN

5. D_U = D_U + MC – F_Avg;

6. ELSE

7. D_U = D_U + MC – Sobol_Var(REMOVE(nth(i U) SobolDecomp(nth(i U) nil nil nil) – F_Avg;

8. RETURN D_U;

Optimization is done by computing Sobol' indices of all circuit parameters with equal weights. Each sensitivity index for a set of parameters S_u measures the uncertainty of interactions of these parameters on circuit specs. In each iteration i, 2^{d-1} number of indices S_{u_i} are calculated constituting the combinations of parameters interactions in the set u. Let ${S_{u_i}}^G$ denote the total sensitivity index for each specification per set of parameters u. In order to decide on the best parameters which contributes to the enhancement of circuit specifications, a cost function S_i is to be determined. The cost function S_i computes the highest effect of set of parameters u on the all circuit specifications. The cost function S_i for m specifications (objectives) for each iteration i is given by:

$$S_i = \max_u \left(\sum_{j=1}^{m} {S_{u_i}}^j \right) \qquad (2.64)$$

The algorithm rejects any solution that tends to change constraints outside the given their boundaries.

Since the problem deals with multi-objective function, in which optimal solution corresponding to each objective is not feasible, the goal is to find a Pareto-optimal set. The most significant parameters, which contribute to the highest output variances of output specs, are optimized to achieve a Pareto-frontier curve. Since the design space varies each step, Sobol' indices are computed in every iteration. If a Pareto-optimal solution is reached, the condition after which it is impossible to achieve higher spec without deteriorating others, a globally non-dominated solution is considered to be attained.

3.Layout Floorplan

Morata *et al.* (1996) introduced an elegant representation of block placement called sequence pair for general floorplans (SP). Like TCG and BSG, but unlike O-tree, B*tree, and CBL, SP is P-admissible. Unlike SP, TCG supports incremental update during operation and keeps the information of the boundary modules as well as their relative positions in the representation. Block placement algorithms that are based on SP use heuristic optimization algorithms, e.g., simulated annealing where generation of large number of sequence pairs are required. Therefore a fast algorithm is needed to generate sequence pairs after each solution perturbation.

3.1. Comments on TCG-S Representation

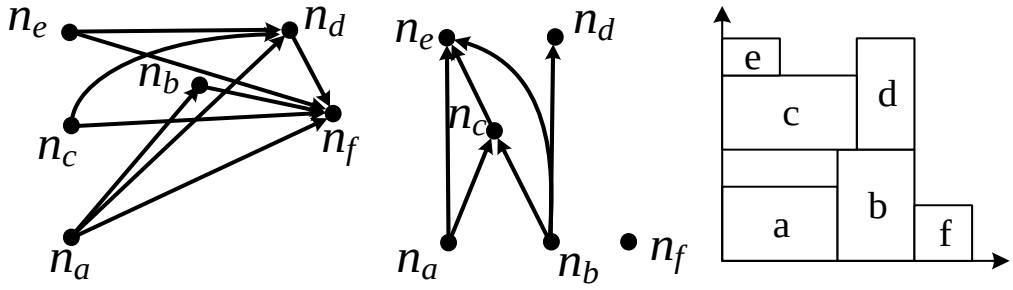
Lin *et al*. proposed a representation which uses the horizontal and vertical transitive closure graphs as well as Γ_- of SP to represent a placement. Based on Γ_- as well as horizontal and vertical contours R_h and R_v, O(n log n) time packing scheme is obtained by sorting and keeping the coordinates of the right (top) boundaries of module in the search order of the Red-Black tree T_h (T_v) [19]. An O(n) runtime packing sequence update was proposed during solution perturbation. The topological ordering of C_h and C_v as well as sequence Γ_- are required to be changed to conform with the new placement under each of the four operations; *rotation, swap, reverse,* and *move*.

Although the three feasibility properties of TCG mentioned in [12] were maintained, they are not sufficient to guarantee an updated configuration of TCG graphs and Γ_- sequence which exactly corresponds to the new placement after each solution perturbation. The TCG-S tuples update algorithm would only be sufficient if the modules subjected to one of the four operations have exactly the same width and length. However, such condition may be satisfied for special constraint placement, e.g., proximity, interdigitated, and common centroid symmetry constraints. The algorithm proposed did not consider geometry of the modules with respect to each other during operations. Therefore, may result in discrepancies between horizontal (vertical) geometric relations of the modules and

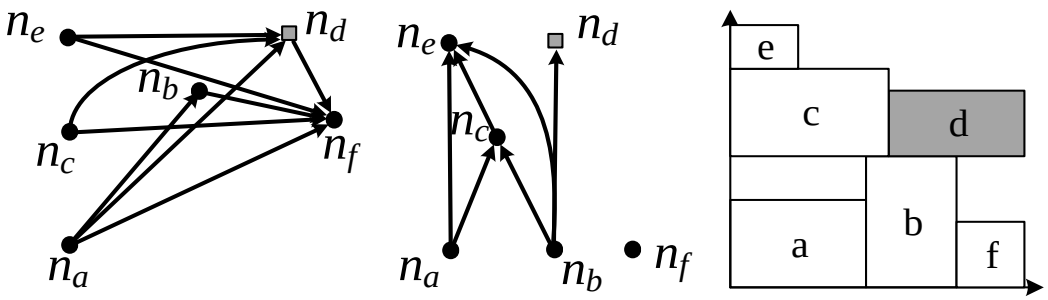
the ones designated by $C_h(C_v)$. Post perturbation on modules b_i and b_j, $b_j \perp b_k$ may accidently be updated as $b_j \vdash b_k$ according to the geometric relation between b_i and b_k. Also $b_k \perp b_j$ through b_i will not be updated in C_v upon swapping b_i and b_j. Edge (b_k, b_j) in C_v will not be deleted and hence, the packing sequence $\Gamma_$will also be incongruously updated. The mismatch between TCG-S representation and its placement will not only lead to non-optimal solution after a series of operations, it may also generate overlapping modules leading to infeasible solution.

In this section, limitations of TCG-S tuples update algorithm are discussed for each operation. Effect of such discrepancy between representation and its corresponding placement on the packing evaluation along with the convergence to the optimal solution will be outlined. Furthermore, a new simple and efficient *O(n)* runtime algorithm for fast realization of incremental update for cost evaluation. The algorithm integrates SP and TCG advantages into TCG-S* a superior topology update scheme which facilitates the search for optimum desired floorplan. Experiments show that TCG-S* is better than existing works in terms of area utilization, stability, and convergence speed.

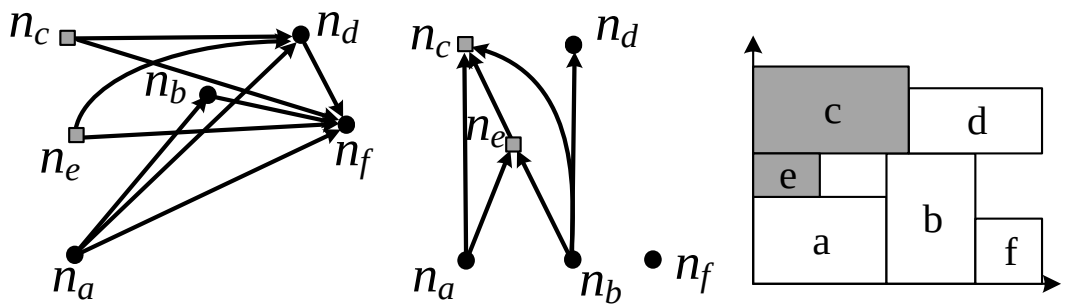
3.1.1. Update of Constraints graph



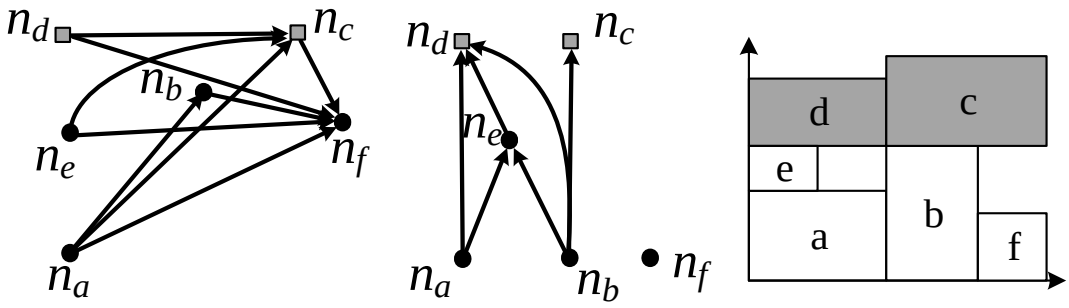
(a) initial configuration of TCG



(b) rotate module d



(c) reverse (n_c, n_e)



(d) swap n_c, n_d

Figure 3.1. Three types of perturbations. (a) The initial TCG (C_h and C_v) and the placement. Dimensions for the six blocks are: a (6 x 4), b (4 x 6), c (7 x 4), d (6 x 3), e (3 x 2), and f (3 x 3). (b) The resulting TCG and placement after rotating module d based on TCG-S. (c) The resulting TCG and placement after reversing nodes $n_c$and n_e based on TCG-S. (d) The resulting TCG and placement after swapping nodes $n_c$and n_d based on TCG-S.

Figure 3.1(a) shows the initial configuration of TCG and its corresponding placement. Module d is rotated as shown in Fig. 3.1(b) and, according to TCG-S, only the weights of the corresponding node d in C_h and C_v are exchanged. Although such an operation has O(1) runtime complexity, it did change the topology of the C_h and C_v, prompting a mismatch between TCG and the corresponding placement. Placement shows that edge (n_d, n_f) should be deleted from C_h and a new edge (n_f, n_d) is to be drawn from node f to node d in C_v.

Figure 3.1(c) shows a reverse operation between two modules c and e. Reverse operation involves reversing the direction of a reduction edge (n_c, n_e) in a transitive closure

graph, which corresponds to deleting edge (n_c, n_e), adding a new edge (n_e, n_c) in the same transitive closure graph C_v. According to TCG-S, for each node $n_k \in$ *fanin*(n_e) $\cup \{n_e\}$ and $n_l \in$ *fanout*(n_c) $\cup \{n_c\}$ in the new graph, the edge (n_k, n_l) is to be added to the graph and the corresponding edges (n_k, n_l) (or(n_l, n_k)) is to be deleted in the other transitive closure graph to maintain the TCG. Therefore, for each node $n_k \in \{a, b, e\}$ and $n_l \in \{c\}$, edge (n_k, n_l) is checked whether it exists in C_v. Since all the edges already exists except (n_e, n_c), nothing is changed. Geometric relation between module b and module e has changed as shown in the placement. Prior the reverse operation, $b_b \perp b_e$, whereas post the reverse, $b_e \vdash b_b$. Consequently, the edge (n_b, n_e) is to be deleted from C_v and a corresponding edge (n_e, n_b) is to be added to the other transitive graph C_h. Since there are at most O(n) n_k's nodes and O(n) n_l's nodes, i.e., O(n^2) (n_k, n_l) edges, time complexity of the reverse operation is O(n^2) where n is the number of modules in a placement,

Figure 3.1(d) shows a TCG and its corresponding placement post swapping module c and d. According to TCG-S, in order to swap two modules c and d, only nodes n_c and n_d designating the modules are to be exchanged in both C_h and C_v. Notice that nodes n_c and n_d have been exchanged in Fig. 3.1(d), where *fanin*(n_c) is exchanged with *fanin*(n_d). Similarly, *fanout*(n_c) is exchanged with *fanout*(n_d). *fanin*(n_c) are $\{n_b\}$ and *fanin*(n_d) are $\{n_a, n_e, n_b\}$. The placement shows that there is no geometric relation between modules b and d in C_v, but rather in C_h. The edge (n_b, n_d) is to be deleted form C_v and a corresponding edge (n_d, n_b) is to be added to the other transitive closure graph C_h.

As a deduction, all operations are prone to changing the topology of the TCGs. The reason of such incongruousity between the TCG and its placement is that the geometry and dimensions of the blocks in a placement with respect to each other has not been considered while perturbing a placement solution.

3.1.2. Packing Sequence Γ_- Update

Consider the TCG and placement shown in Fig. 3.1(c). The packing sequence Γ_- can be obtained using equivalence of SP and TCG proposed by [18], by repeatedly extracting a node n_i with *fanin*(n_i) = 0 in $C_h$and C_v. Similarly, Γ_+ is obtained by repeatedly extracting

a node n_i with *fanin(n_i)* = 0 in C_h and *fanout(n_i)* = 0 in C_v. Accordingly, the sequences Γ_+ and Γ_- are (⟨c e a d b f⟩, ⟨a b e c d f⟩) respectively. For evaluating SP, packing cost can be calculated using the longest common sequence proposed by [11]. By computing lcs(Γ_+, Γ_-) and lcs(Γ_+^R, Γ_-), width and height are determined and hence, the whole placement area. The positions of the modules during each solution perturbation can be computed while evaluating the packing cost using the last common sequence algorithm. Based on C_v graph shown in Fig. 3.1(c) and the aforementioned LCS algorithm, $y_e > y'_b$. lcs(Γ_+, Γ_-) which holds the value of block f position plus its weight in x-direction (y'_f), equals to 16. Whereas, lcs(Γ_+^R, Γ_-), which holds the value of block c position plus its weight in y-direction, equals to 12. Placement span in the initial TCG configuration is (13, 12), became (16, 12) after reverse operation. Thus the perturbing solution is diverging and deviating from the desired one. The mismatch between TCG and its corresponding placement during perturbation is obvious.

Lin *et al*. proposed a scheme for updating sequence Γ_- in reverse operation, in which module b_i is deleted and inserted following b_j in sequence Γ_-. For each module b_k between b_i and b_j in the sequence Γ_-, in which edge (n_i, n_k) exists in the graph, b_k is deleted and inserted following the most recently inserted module. Consider the placement shown in Fig. 3.1(b), Assume that edge (n_a, n_e) is reversed. Edges (n_a, n_k), where node $n_k \in \{n_c, n_b, n_e\}$ and node $n_l \in \{n_a, n_c\}$, that doesn't exist in the C_v graph will be added to C_v and deleted from the corresponding graph. Therefore, the new added edges are (n_c, n_a), (n_e, n_c), (n_b, n_a), and (n_e, n_a). Accordingly, b_a is deleted from Γ_- and inserted following b_e. Since, edge (n_a, n_c), (n_a, n_b) doesn't exist nothing is changed. The new Γ_- is ⟨b c e a d f⟩, whereas transforming TCG into SP results in Γ_- equals ⟨b e c a d f⟩. Thus, the proposed algorithm for updating Γ_- is only feasible if the edge considered for move is a reduction edge, where no module b_k exists between b_i and b_j. Incongruous TCG graphs and its corresponding Γ_- results in infeasible solution during packing cost evaluation by the binary search tree.

Therefore, the limitations of the proposed update scheme in [18] did not only tend to increase the convergence time of the floorplan and make it harder to converge to the desired

solution, by miscalculating packing cost, it may also generate infeasible solution after a series of operations.

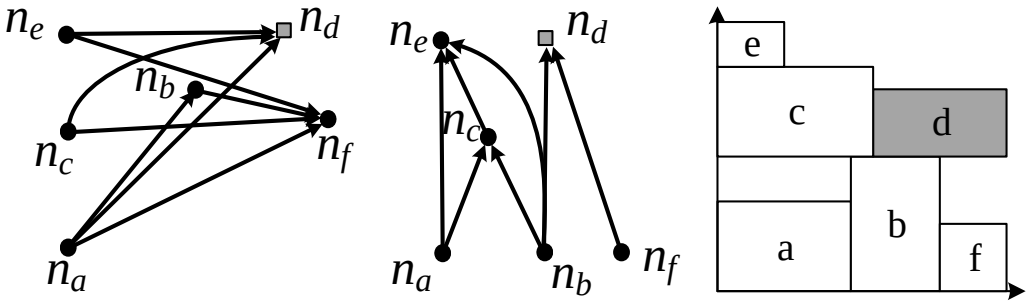
3.2. TCG-S* Perturbing Algorithm

3.2.1. TCG Topology Update

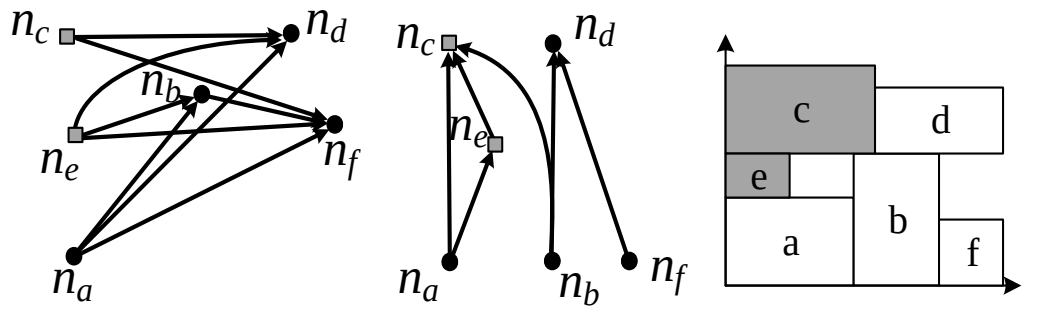
The section proposes a new simple and efficient O(n) runtime algorithm, where n is the number of modules in a placement, for the update of the constraint graphs C_h and C_v during perturbation, based on the knowledge of the position of the modules.

3.2.1.1. Rotate

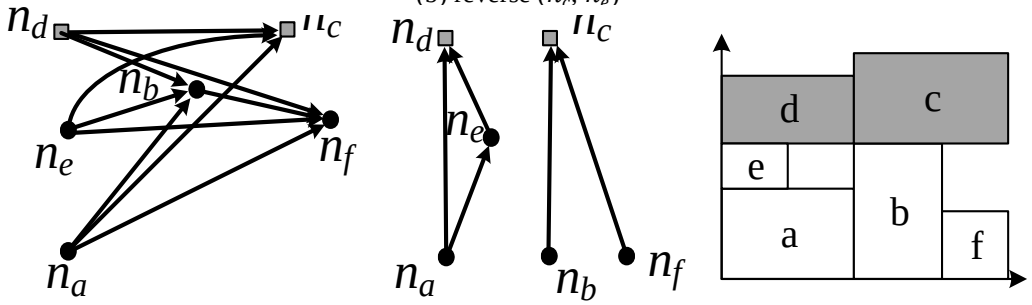
The rotate operation involves rotating a module b_i without changing its position. Rotate operation involves exchanging weights of module b_i in both C_h and C_v. Edges (n_i, n_k) are required to be updated in both C_h and C_v, where $n_k \in$ *fanin*(n_i) $\cup$ *fanout*(n_i) in both C_h and C_v. First, edge (n_i, n_j) is deleted from C_v and added to C_h, where $n_j \in$ *fanout*(n_i) $\cup$ *fanin*(n_i). All modules $b_j \in$ *fanout*(n_i) in $C_h \cup$ *fanout*(n_i) in C_v in which $y_j > y_i$ are checked whether there exists a vertical relation with b_i. If exists, an edge (n_i, n_j) is added to C_v and the corresponding edge (n_i, n_j) is deleted from the other transitive graph. Otherwise, an edge (n_i, n_j) is added to C_h. Similarly, to obtain *fanin*(n_i) in C_v and its corresponding update in both C_h and C_v, all modules b_j with $y'_j < y'_i$ are checked whether there exists a vertical relation with b_i. If exists, an edge (n_j, n_i) is added to C_v and the corresponding edge (n_i, n_j) is deleted from C_h. Otherwise, edge (n_i, n_j) is added to C_h.



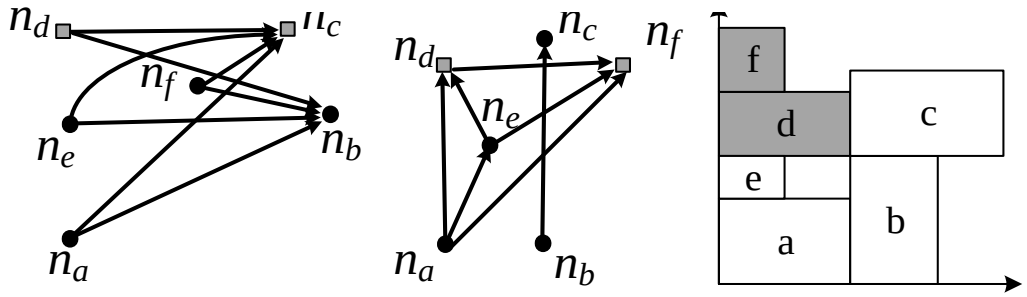
(a) rotate module d



(b) reverse (n_c, n_e)



(c) swap n_c, n_d



(d) move n_d, n_f

Figure 3.2. Three types of perturbations. (a) The resulting TCG and placement after rotating module. (b) The resulting TCG and placement after reversing nodes $n_c$and n_e. (c) The resulting TCG and placement after swapping nodes $n_c$and n_d.

Figure 3.2(a) shows the resulting TCG and its corresponding placement post reversing module d. Notice that weights of the node n_d have been exchanged in both C_h and C_v. *fanout*(n_i) ∪ *fanin*(n_i) = {n_b}. Therefore, edge (n_d, n_j) is deleted from C_v and added to C_h, where $n_j \in \{n_b\}$. *fanout*(n_d) in $C_v = \emptyset$, *fanout*(n_d) in $C_h = \{n_f\}$. Since module $y_f < y_d$, nothing is changed. To obtain *fanin*(n_d) in C_v, module $n_j \in \{n_b, n_f\}$ in which $y'_j < y'_d$ and n_j has vertical relation with module d, is added to C_v and the corresponding edge (n_d,

n_j) is to be deleted from C_h. Therefore, edges (n_b, n_d) and (n_f, n_d) are added to C_v and edge (n_d, n_f) is deleted from C_h.

Theorem 1: Rotate operation takes O(n) runtime, where n is the number of modules in a placement.

Proof: The time complexity is dominated by checking whether $n_i \perp n_j$, where $n_j \in$ *fanout*(n_i) in $C_h \cup$ *fanout*(n_i) in C_v, and by deleting all edges (n_i, n_k) from C_v, where $n_k \in$ *fanout*(n_i) $\cup$ *fanin*(n_i). Since there are at most O(n) n_j's and O(n) n_k's, rotate operation only takes O(n) runtime in total.

3.2.1.2. Swap

To swap modules b_i and b_j, their values in the position array are exchanged. Edge (n_i, n_j) is deleted from a transitive closure graph and a corresponding edge (n_j, n_i) is added to the same graph. Edges (n_k, n_j), where node $n_k \in$ *fanin*(n_j) $\notin$ *fanin*(n_i) in C_h, are deleted from C_h and corresponding edges (n_k, n_i) are added to C_h. Similarly, Edges (n_i, n_k), in which node $n_k \in$ *fanout*(n_i) $\notin$ *fanout*(n_j) in C_h, are deleted from C_h and corresponding edges (n_j, n_k) are added to C_h. Edges (n_j, n_k), where node $n_k \in$ *fanout*(n_j) $\notin$ *fanout*(n_i) in C_v, are deleted from C_v. Similarly, edges (n_i, n_k), where node $n_k \in$ *fanout*(n_i) $\notin$ *fanout*(n_j) in C_v, are deleted from C_v. Edges (n_j, n_k), where node $n_k \in$ *fanin*(n_j) $\notin$ *fanin*(n_i) in C_v, are deleted from C_v. Similarly, edges (n_i, n_k), where node $n_k \in$ *fanin*(n_i) $\notin$ *fanin*(n_j) in C_v, are deleted from C_v. For nodes $n_k \in$ *fanout*(n_j) in $C_v \cup$ *fanout*(n_j) in C_h, where $x'_k > x_i$ and $b_i \perp b_k$, an edge (n_i, n_k) is added to C_v. If else, edge (n_i, n_k) is added to C_h. Similarly, for nodes $n_k \in$ *fanout*(n_i) in $C_v \cup$ *fanout*(n_i) in C_h, where $x'_k > x_j$ and modules b_k and b_j exhibits a vertical geometric relation, an edge (n_j, n_k) is added to C_v. If else, edge (n_j, n_k) is added to C_h. For nodes $n_k \in$ *fanin*(n_j) in $C_v \cup$ *fanout*(n_j) in C_h, where $x'_k > x_i$ and modules b_k and b_i exhibits a vertical geometric relation, an edge (n_i, n_k) is added to C_v. If else, edge (n_i, n_k) is added to C_h. Similarly, for nodes $n_k \in$ *fanin*(n_i) in $C_v \cup$ *fanout*(n_i) in C_h, where $x'_k > x_j$ and modules b_k and b_j exhibits a vertical geometric relation, an edge (n_j, n_k) is added to C_v. If else, edge (n_j, n_k) is added to C_h.

Figure 3.2(c) shows the resulting TCG and its corresponding placement after swapping modules b_c and b_d. Notice that their positions have been exchanged. Edge (n_c, n_d) is deleted from C_h and a corresponding edge (n_d, n_c) is added to C_h. *fanin*(n_d) in C_h = {n_e, n_a}, *fanin*(n_c) = {$\emptyset$}. Therefore, edges (n_e, n_d) and (n_a, n_d) are deleted from C_h and corresponding edges (n_e, n_c) and (n_a, n_c) are added to C_h, where nodes n_e and $n_a \in$ *fanin*(n_d) $\notin$ *fanin*(n_c). *fanout*(n_c) = {n_f}, *fanout*(n_d) = {$\emptyset$}. Accordingly, edge (n_c, n_f) is deleted from C_h and corresponding edge (n_d, n_f) is added to C_h. Since *fanout*(n_d) in C_v = {$\emptyset$} and *fanout*(n_c) = {$\emptyset$}, *fanout* of nodes n_d and n_c in C_v is not changed. *fanin*(n_d) = {n_b, n_f}, *fanin*(n_c) = {n_a, n_b, n_e}. Edge (n_f, n_d) is deleted from C_v, where $n_f \in$ *fanin*(n_d) $\notin$ *fanin*(n_c). Similarly, edges (n_e, n_c) and (n_a, n_c) are deleted from C_v, where nodes n_a and $n_e \in$ *fanin*(n_c) $\notin$ *fanin*(n_d). Since *fanout*(n_d) in $C_v \cup$ *fanout*(n_d) in C_h = {$\emptyset$}, *fanout*(n_c) in C_h is not changed. Since *fanout*(n_c) in $C_v \cup$ *fanout*(n_c) in C_h = {$\emptyset$}, *fanout*(n_d) in C_h (C_v) is not changed. *fanin*(n_d) in $C_v \cup$ *fanout*(n_d) in C_h = {n_b, n_f}. Edge(n_f, n_c) is added C_v as modules $b_f \perp b_c$. Similarly, edges (n_a, n_d) and (n_e, n_d) are added to C_v.

Theorem 2: Swap operation takes O(n) runtime, where n is the number of modules in a placement.

Proof: The time complexity is dominated by checking whether $n_i \perp n_k$ ($n_j \perp n_k$), where $n_k \in$ *fanout*(n_j) (*fanout*(n_i)) in $C_v \cup$ *fanout*(n_j) (*fanout*(n_i)) in C_h, and checking whether $n_i \perp n_l$ ($n_j \perp n_l$), where $n_l \in$ *fanin*(n_j) (*fanin*(n_i)) in $C_v \cup$ *fanout*(n_j) (*fanout*(n_i)) in C_h. Since there are at most O(n) n_k's and O(n) n_l's, operation takes O(n) runtime in total.

3.2.1.3. Reverse

Reverse operation reverses the geometric relation between two modules b_i and b_j. If there exists a geometric relation $b_i \vdash b_j$, the new relation after reversing is $b_j \vdash b_i$.

Reverse operation is a derivative of swap operation, since it involves reversing the direction of an edge (n_i, n_j), i.e. swap modules b_i and b_j. Hence, TCG topology update in a reverse operation only Swap operation on block b_j.

3.2.1.4. Move

Move operation involves changing the geometric relation of two modules b_i and b_j between horizontal transitive closure graph and vertical one. The move operation can be classified into two instances, the one where $b_i \perp b_j$, and the other where $b_i \vdash b_j$.

To move an edge (n_i, n_j) in C_h (C_v), edge (n_j, n_k) is deleted from C_v, where module $n_k \in$ *fanout*(n_j). Edge (n_k, n_j) is deleted from C_v, where module $n_k \in$ *fanin*(n_j). Edge (n_j, n_l), where $n_l \in$ *fanout*(n_j), is deleted from C_v. For each node $n_k \in$ *fanout*$(n_j) \cup$ *fanin*(n_j) in $C_v \cup$ *fanin*(n_j) in C_h, if $b_j \perp b_k$ or $b_k \perp b_j$, then edge (n_k, n_j) $((n_j, n_k))$ is deleted from C_h. If $b_j \perp b_k$, then edge (n_j, n_k) is added to C_v. Else, edge (n_k, n_j) is added to C_v. If no geometric vertical relation exists between modules b_j and b_k and $b_j \vdash b_k$ ($b_k \vdash b_j$) in x-direction, then edge (n_k, n_j) $((n_j, n_k))$ is deleted from C_h and a corresponding edge (n_j, n_k) $((n_k, n_j))$ is added to C_h. To update *fanout* of node n_i in C_h and C_v, For each node $n_k \in$ *fanout*(n_i) in $C_v \cup$ *fanout*(n_i) in C_h and $y_k > y_i$. If $n_i \perp n_k$ or $n_k \perp n_i$, then edge (n_i, n_k) is deleted from C_h and the corresponding edge (n_i, n_k) is added to C_v. If no vertical relation exists between modules b_i and b_k, then the edge (n_i, n_k) is deleted from C_v and the edge (n_i, n_k) (or(n_k, n_i)) is added to C_h.

Figure 3.2(d) shows the resulting TCG and its corresponding placement after moving the edge (n_d, n_f) in the C_h in Fig. 3.2(c) to C_v. *fanout*(n_f) in $C_v = \{n_c\}$, *fanin*(n_f) in $C_h = \{n_a, n_b, n_d, n_e\}$, and *fanin*$(n_f)$ in $C_v = \{\emptyset\}$. Consequently, edge (n_f, n_c) is deleted from C_v. *fanout*$(n_f) \cup$ *fanin*(n_f) in $C_v \cup$ *fanin*(n_f) in $C_h = \{n_a, n_b, n_c, n_d, n_e\}$. Since modules b_a, b_e, and $b_d \perp b_f$, where $\{n_a, n_d, n_e\} \subset \{n_a, n_b, n_c, n_d, n_e\}$, edge (n_a, n_f), (n_d, n_f), (n_e, n_f) is deleted from C_h and corresponding edges are added to C_v. Edges (n_b, n_f) is deleted from C_h and edges (n_f, n_b) and (n_f, n_c) are added to C_h. *fanout*(n_d) in $C_v \cup$ *fanout*(n_d) in $C_h = \{n_c, n_b, n_f\}$, from which only $b_d \perp b_f$. Therefore, edge (n_d, n_f) is checked whether it exists in C_h. Since edge (n_d, n_f) does not exist in C_h and edge (n_d, n_f) has already been added to C_v, nothing is done.

Theorem 3: Move operation takes O(n) runtime, where n is the number of modules in a floorplan.

Proof: The time complexity is dominated by checking whether $b_j \perp b_k$ ($b_k \perp b_j$), where $n_k \in$ *fanout*(n_j) $\cup$ *fanin*(n_j) in $C_v \cup$ *fanin*(n_j) in C_h, and checking $b_i \perp b_l$ ($b_l \perp b_i$), where $n_l \in$ *fanout*(n_i) in $C_v \cup$ *fanout*(n_i) in C_h. Since there are at most O(n) n_k's and O(n) n_l's, the operation takes O(n) in total.

Theorem 5: No reduction edges are required to be obtained for Swap, Reverse and Move operations.

Proof: An edge (n_i, n_j) is considered a reduction edge if there does not exist another path from n_i to n_j except the edge (n_i, n_j). Swap, Reverse and move perturbations do not require to operate only on reduction edges as in TCG-S representation, since operations in TCG-S* update the closure edges (n_i, n_j) along with all the reduction edges that form other paths from n_i to n_j. Therefore, the resulting TCGs are acyclic. Operating on both reduction and closure edges increase available move combinations, and facilitates the search for minimum packing cost, i.e. the desired solution.

Property 4: *fanin* (*fanout*) edges in C_v and *fanin* edges in C_h must be acyclic.

To guarantee feasible TCG, edges drawn from node n_i to n_j in the *fanout* (n_k to n_i in the *fanin*) of C_v, as of geometric relation between modules b_i and b_j $b_i \perp b_j$, and edges drawn from node n_k to n_j in the *fanin* of C_h as $b_k \vdash b_j$ must be acyclic. Since acyclic edges in C_h (C_v) does not guarantee a feasible solution, nodes n_i, n_j, and n_k must be checked that their edges in C_v and C_h combined are acyclic. $b_i \perp b_j$ ($b_i \vdash b_j$), $b_k \perp b_i$ ($b_k \vdash b_i$), and $b_k \vdash b_j$ ($b_k \perp b_j$) cannot exist in a TCG, and thus edges (n_i, n_j), (n_k, n_i) in C_h (C_v) and (n_k, n_j) in C_v (C_h) cannot exist.

3.2.2. Packing Sequence Update

This section introduces an O(n) runtime algorithm, where n is the number of modules in a placement, for the update of packing sequences Γ_+ and Γ_- based on knowledge of C_h, C_v,

and the positions of the modules. The algorithm depends on updating the TCG topology after each perturbation.

Algorithm 3.1: Update-SP (SeqX, SeqY, A)

```
//initialize SeqYNew Arrays with 0
//initialize Tmp List with nil
1.    FOR i 0 NUM(SeqY)-1
2.        IF( SeqY[i] ∈ Fout_Cv(A) in C_v ∪ Fout_Ch(A) THEN {
3.            SeqYNew[i]=SeqY[i];
4.        ELSE
5.            Tmp = concat{Tmp SeqY[i]}; }
6.    FOR i NUM(SeqY)-NUM(Tmp) NUM(SeqY)-1
7.        SeqYNew[i] = nth(i-NUM(SeqY)+NUM(Tmp) Tmp);
8.    Tmp = nil;
9.    RETURN SeqYNew
```

Algorithm 1 shows the update of Γ_-, sequence Γ_+ update will be discussed shortly. The algorithm updates the position of the module b_i, on which perturbation is applied, with respect to the ones that precedes and the ones that follows it in the sequence. Any module b_k, belongs to *fanout(b_i)* in C_v graph ∪ *fanout(b_i)* in C_h graph, is to follow module b_i in the sequence Γ_-. When the algorithm ends, the array SeqYNew[1…n] records the sequence Γ_-. Similarly, to update sequence Γ_+, Any module b_k, belongs to *fanin(b_i)* in C_v graph ∪ *fanout(b_i)* in C_h graph, is to follow module b_i in the sequence Γ_+.

Tang *et al.* proposed a fast packing cost evaluation of sequence pair by computing the longest common subsequence with minimum time complexity of O(n log log n). However, time complexity of the floorplan algorithm is dominated by the construction of constraint graphs from scratch after each perturbation for packing cost evaluation, since the geometric relations between modules are not transparent to the operations of SP. Thus, the time complexity of constructing the constraint graphs is O(n^2), where n is the number of modules in a placement. Implementing TCG-S* algorithm with O(n) runtime in total

decreases the time complexity of the sequence pair floorplan algorithm to O(n log log n) for significantly large n.

Theorem 5: Algorithm 3.1 correctly returns the new sequence pairs Γ_+ and Γ_-.

Proof: According to sequence pair representation, packing sequence Γ_- is constructed by concatenating the nodes in a placement as in (1) and (2) subject to the condition that either b_i is left to or below b_j, where b_j follows b_i in the sequence. Therefore, b_j follows b_i in Γ_- only if $b_i \vdash b_j$ or $b_i \perp b_j$. Additionally, based on property (2) of TCG discussed in [2], the two nodes n_i and n_j are connected by exactly one edge either in C_v or C_h. If $n_j \notin$ *fanout*(b_i) either in C_v or C_h, then $n_j \in$ to *fanin*(b_i) either in C_v or C_h. Therefore, algorithm 3.1 correctly returns the new sequence pair.

Theorem 6: Algorithm 3.1 updates the packing sequences in O(n) runtime.

Proof: The time complexity of updating sequence Γ_- in algorithm 3.1 is dominated by checking whether b_j is a member of *fanout(b_i)* in both C_v and C_h. Since, time complexity of updating sequence Γ_+ and Γ_- are the same, and in worst case scenario there are at most O(n-1) of b_j's, time complexity of algorithm 1 is O(n) in total.

3.2.3. Equivalence of TCG and SP

Lin *et al.* proposed a transformation from TCG to SP using *fanin* and *fanout* of TCGs [18]. Time complexity of such algorithm merely depends on the configuration of TCG. For each node n_k in the TCG, a node n_l is checked whether edge (n_k, n_l) or (n_l, n_k) exists in C_h or C_h. if exists, the edge is deleted. In worst case, there exist O(n-1) n_k's and O(n) n_l's, thus the time complexity is O(n^2). TCG-S* packing sequence update algorithm returns the updated sequences Γ_+ and Γ_- in O(n) runtime which makes it superior to the update proposed by [18].

Likewise, a reverse transformation from SP to TCG can be obtained. Given a sequence pair (Γ_+, Γ_-), the *fanin* and *fanout* of all nodes in both transitive closure graphs can be obtained by determining the common nodes in the subsequence of the inspected node in each of Γ_+ and Γ_- according to the horizontal and vertical constraints. Accordingly, in order to obtain *fanout*(n_i) in x-direction from Γ_+ and Γ_-, subsequence of node n_i in $\Gamma_+ \cap$

subsequence of node n_i in Γ_- is determined. Subsequence of node n_i in Γ_+^R ∩ subsequence of node n_i in Γ_- determines *fanout*(n_i) in y-direction. Subsequence of node n_i in Γ_+^R ∩ subsequence of node n_i in Γ_-^R determines *fanin*(n_i) in x-direction. Finally, subsequence of node n_i in Γ_+ ∩ subsequence of node n_i in Γ_-^R determines *fanin*(n_i) in y-direction. Example, for the placement shown in Fig. 3.2(a) with sequence pair (⟨e c a d b f⟩, ⟨a b c e f d⟩), *fanout*(n_a) in x-direction = {n_b, n_d, n_f}, *fanout*(n_a) in y-direction = {n_c, n_e}, *fanin*(n_a) in x-direction = {∅}, and *fanin*(n_a) in y-direction = {∅}.

3.3. Floor Planning Algorithm

A simulated annealing based algorithm [54] is developed using TCG-S for non-slicing floorplan design with the updated perturbing algorithm TCG-S*. Given an initial solution represented by TCG and SP, the algorithm perturbs the placement to obtain new TCG and SP. The new TCG must satisfy the three properties mentioned in [12], and the new packing sequences pair must show equivalence with TCG as well. Slack computation proposed by [55] is implemented in order to improve move selection in simulated annealing. Contribution to wirelength minimization is discussed in this section as well.

3.3.1. Slack Computation

Blocks that constrain each other in the same direction in the order that any attempt to minimize path length will result in blocks overlap, lie on the critical path of floorplan. Hence, the slack value in that direction is zero. These blocks are good candidates for move selection towards reducing span of the floorplan. Slack based moves along with the moves of TCG give a directed movement towards area minimization through the determination of zero slack blocks, which represents the critical paths of floorplan.

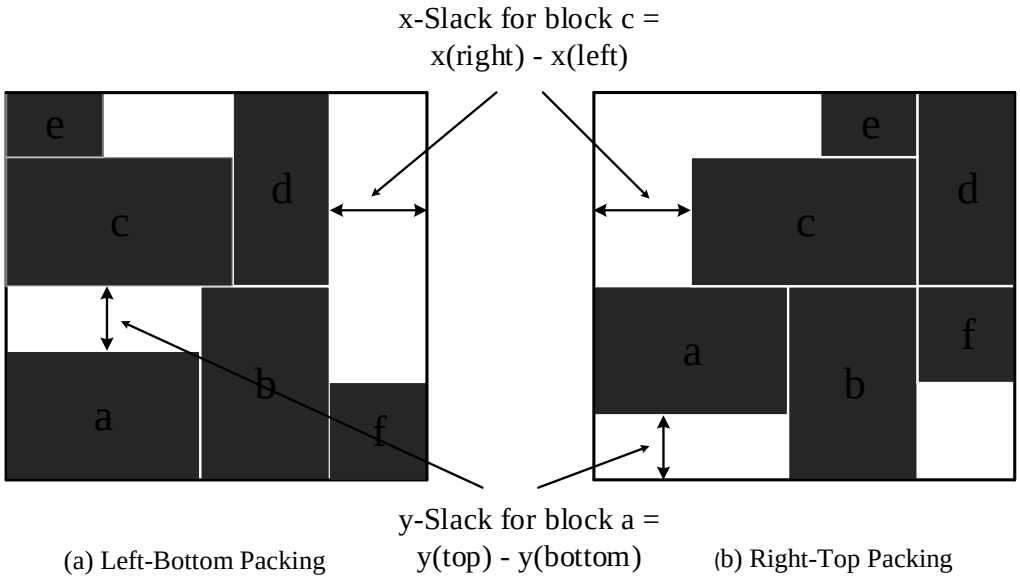


Figure 3.3. Slack computation (a) floorplan evaluation in left to right and bottom to top mode. (b) floorplan evaluation from right to left and top to bottom mode.

Table 1. MCNC Benchmark circuits

Circuit	#Module	#I/O Pads	#Nets	#Pins
apte	9	73	97	214
xerox	10	107	203	696
hp	11	43	83	264

Slacks can be computed in left-to-right mode or right-to-left mode. Fig. 3.3 shows floorplan evaluation for the same sequence pair in bottom-left mode and top-right mode.

To compute slacks of blocks in floorplan, first, LCS of the two sequences is computed in the left to right mode. Then the two sequences are reversed for LCS computation is the left to right mode. For example, LCS of blocks in x-direction in the left to right mode is computed by calculating lcs(Γ_+^R, Γ_-^R), whereas to compute LCS in y-direction, lcs(Γ_+, Γ_-^R) is calculated. Algorithm 3.2 computes the LCS of the blocks using the sequence pair. Algorithm 3.3 calls LCS function after initializing the sequence pair in reversed order.

Algorithm 3.2:

LCS_Calc(X,Y, weights)

1. initialize_length_array L with 0;
2. initialize_position_array P;
3. initialize_result_array R;
4. For i = 0 TO n-1 DO
5. p = match[X[i]];
6. b = X[i];
7. max = L[p]+weights[i];
8. P[i] = L[p];
9. For j = p TO n-1 DO
10. IF(max > L[j] && Y[j] ∈ Fout(b))
11. THEN
12. L[j] = max;
13. R[0] = P[0,…,n-1];
14. R[1] = L[n-1];
15. RETURN R;

Algorithm 3.3:

Slack (X,Y, PosX, PosY, wX, wY)

1. initialize_arrays Rx_BL, Ry_BL;
2. initialize_array Rx_TR, Ry_TR;

3. /*evaluate LCS X in bottom-left mode*/

4. LCSX_BL = LCS_Calc(X,Y, wX);

5. /*evaluate LCS Y in bottom-left mode*/

6. For i = 0 TO n-1 DO

7. X^R[i] = X[n-1-i];

8. WY_{BL}^R[i] = wY[n-1-i];

9. LCSY_BL = LCS_Calc(X^R, Y, WY^R);

10. /*evaluate LCS X in top-right mode*/

11. For i = 0 TO n-1 DO

12. Y^R[i] = Y[n-1-i];

13. WX^R[i] = wX[n-1-i];

14. $LCSX^R$_TR = LCS_Calc(X^R, Y^R, WX^R);

15. /*evaluate LCS Y in top-right mode*/

16.$LCSY^R$_TR = LCS_Calc(X, Y^R, wY);

17. For i = 0 TO n-1 DO

18. LCSX_TR[i] = $LCSX^R$_TR[n-1-i];

19. LCSY_TR[i] = $LCSY^R$_TR[n-1-i];

20./*compute slack*/

21. For i = 0 TO n-1 DO

22. SlackX[i] = max(LCSX_BL[i])-LCSX_BL[i]-LCSX_TR[i]+wX[i];

23. SlackY[i] = max(LCSY_BL[i])-LCSY_BL[i]-LCSY_TR[i]+wY[i];

Based on the equivalence between TCG and SP, LCS function returns floorplan span in x-direction (y-direction) faster. Since block b_i in a placement is only bounded by its *fanout* blocks in C_h (C_v), only these blocks affect the total length of candidates sequences in the path of block b_i. Let k denote the index of module b_i in sequence Γ_+ and p denote the index of mobule b_i in sequence Γ_-. Therefore, computing lcs(Γ_+[1…k], Γ_-[1…p]) only considers the *fanout* of blocks in the common subsequence of (Γ_+[1…k-1], Γ_-[1…p-1]).

4.Placement and Routing

4.1. Constraints-based Placement

Placement of analog circuits is an error prone and time consuming process. It can easily take an experienced designer weeks or months to layout even a relatively small circuit. Some devices are needed to be placed at close proximity and symmetrically with respect to an axis or to a center point. This can reduce the effect of parasitic mismatches, which will cause degradation of the circuit performance. Circuit sensitivity to thermal gradients and process variations can be reduced by placing symmetric devices close to each other.

4.1.1. Overview of Analog Placement Methods

In order to automatically produce analog device-level layouts matching in density and performance the high-quality manual layouts, a placement tool must not only provide a good *rectangle packing* functionality (which must be common to any placement method) but, additionally, it must include also analog-specific capabilities. Such specific features are, for instance; 1) the ability to deal with topological constraints for symmetry and device matching; 2) the ability to arrange devices such that critical structures are shared in common (also known as *device merging*) in order to reduce both layout density and induced parasitics; and 3) the existence of a (built-in) library of predefined module generators and the ability to exploit their reshaping capabilities during the placement process. Besides these specific features of analog placement, the main goal of optimally packing arbitrarily sized modules is similar to that of other very large scale integrated circuits (VLSI) placement problems—chip floorplanning, standard cell and macro cell digital placement. Due to the complexity of the basic problem, several heuristic classes of placement techniques have been attempted.

The constructive placement techniques, which consist in evolving gradually the placement solution by selecting one module at a time and positioning it in the “best” available location, were among the first developed for VLSI layout. Several systems for analog placement employ constructive methods: Kayal *et al.* developed an expert

knowledge base to guide the placement [56]; Mehranfar suggested a schematic-driven approach, using a constructive scheme based on connectivity and relative positioning in the input schematic [57]. Although these methods are fast, scaling well with the problem size, the results can be poor due to the order dependence, lacking of global view in dealing with a variety of interacting quality measures. Branch-and-bound placement techniques use a controlled enumeration of all possible layout configurations in the search space, where a lower bound of the chosen cost function is used to prune the search. The branch-and-bound algorithms eventually find the optimal solution as they explore exhaustively the search space. However, they are effective only for problems of very small size as the number of visited configurations grows exponentially with the size of the problem. The related integer linear programming (ILP) placement models suffer the same scaling drawback as most ILP packages are based on branch-and-bound approaches. Even if the placement problems are tackled hierarchically, the branch-and-bound methods are less attractive for analog device placement due to usually a much larger search space than digital problems of similar size (for instance, due to the presence of "soft" capacitors which can be implemented in a large number of versions). More recently, a placement technique iteratively combining min-cut partitioning and force-directed placement (DLP) has been employed in an interactive environment for full-custom designs [58].

The simulated annealing [54] and genetic algorithms are the most effective choice for solving industrial analog placement problems. These algorithms use stochastically controlled hill-climbing to avoid local minima during the optimization process. In addition, they do not impose severe constraints on the size of the problems or on the mathematical properties of the cost function. While efficiently trading off between a variety of layout factors as area, total net length, aspect ratio, maximum chip width and/or height, cell orientation, "soft" cell shape, etc., they are very flexible—supporting incremental addition of new functionality, and they are relatively easy to implement (although good tuning needs more time).

Existing approaches to automated placement generation can be classified into two categories;

i. Template driven layout

This approach is based on a known layout pattern or layout template which specifies necessary device-to-device, device-to-wire, or wire-wire special relationship for a typical circuit. It is fast and easy to obtain a compact layout. However, this approach lacks flexibility as matching varies from circuit design to another.

ii. Constraint-based layout

It is more flexible than template driven layout approach. Fig. 4.1 shows the general flow of the constraint-driven or performance-driven layout. 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 area utilization.

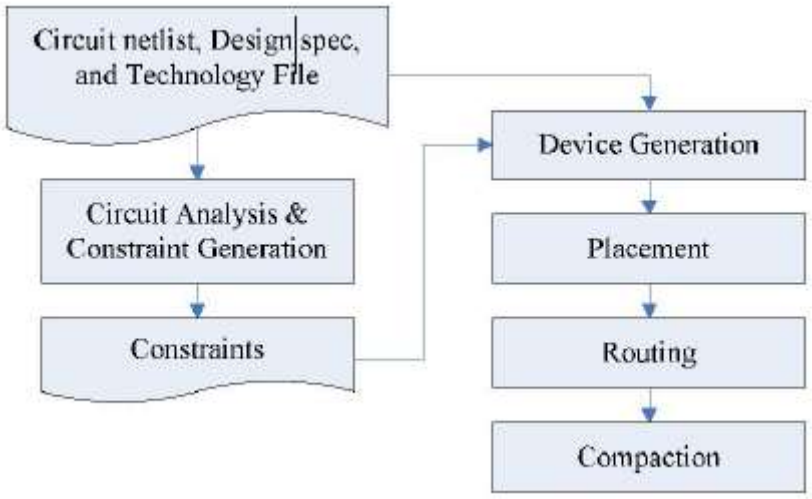


Figure 4.1 Constraint-driven analog layout generation flow

According to [48] and [49], device group placement is classified into four categories; the cross-couple, inter-digitated, common-centroid, and general stacking matching styles. These four styles are studied thoroughly in [50]. This section mainly studies and impements the common-centroid and inter-digitated matching styles in automated device group

placement in order to reduce systematic device mismatch. The inputs of placement algorithm are the aspect ratio bounds, which is computed in the floorplan optimization process, devices to be matched, and matching style.

4.1.2. A Review on Simulated Annealing Optimization Algorithm

At each layout optimization stage, one wants to optimize the eventual performance of the system without compromising the feasibility of the subsequent stage. The basic elements of simulated annealing are:

i. A finite set S.
ii. A real-valued cost function *J* defined on *S*. let S^* be the set of global minima of the function *J*, assumed to be a proper subset of *S*.
iii. For each $i \subset S$, a set $S(i) \subset S - \{i\}$, called the set of neighbors of i.
iv. For every i, a collection of positive coefficients $q_{ij}, j \in S(i)$, such that $\sum_{j \in S(i)} q_{ij} = 1$. It is assumed that $j \in S(i)$ if $i \in S(j)$.
v. A non-increasing function T: $N - [0, \propto]$, called the cooling schedule. N is the set of positive integers, and T(t) is called the temperature at time t.
vi. An initial state $x(0) \in S$.

Given the above elements, the SA algorithm consists of a discrete-time inhomogeneous Markov Chain $x(t)$, whose evolution we now describe. If the current state $x(t)$ is equal to i, choose a neighbor j to i at random; the probability that any particular $j \in S(i)$ is selected is equal to q_{ij}. Once j is chosen, the next state $x(t+1)$ is determined as follows:

If $J(j) \leq J(i)$, then $x(t+1) = j$

If $J(j) > J(i)$ then

$X(t+1) = j$ with probability $exp[-(J(j) - J(i))/T(t)]$

$X(t+1) = i$ otherwise

Formally,

$$P[x(t+1) = j | x(t) = i] = q_{ij} \exp\left[-\frac{1}{T(t)} max\{0, J(j) - J(i)\}\right]$$

$$if\ j \neq i, j \in S(i) \quad (4.1)$$

If $j \neq i, j \notin S(i), then\ P[x(t+1) = j | x(t) = i] = 0.$

The rationale behind the SA algorithm is best understood by considering a homogeneous Markov chain $X_T(t)$ in which the temperature $T(t)$ is held at constant value T. Assume that the Markov chain $X_T(t)$ is irreducible and periodic and that $q_{ij} = q_{ji}$ for all i, j. Then $X_T(t)$ is a reversible Markov chain, and its invariant probability distribution is given by

$$\pi_T(i) = \frac{1}{Z_T} \exp\left[-\frac{J(i)}{T}\right] \ i \in S, \quad (4.2)$$

where Z_T is a normalizing constant. (This is easily shown by verifying that the detailed balance equations hold). The probability distribution function π_T is concentrated on set S^* of global minima J. This latter property remains valid if the condition $q_{ij} = q_{ji}$ is relaxed.

The probability distribution (4.2), known as the Gibbs distribution, plays an important role in statistical mechanics. Statistical physicists have been interested in generating a sample element S, drawn according to the probability distribution π_T. This is accomplished by simulating Markov chain $X_T(t)$ until it reaches equilibrium, where this method is known as Metropolis algorithm (Metropolis et al., 1953). In the context of optimization, an optimal element of S can be generated with high probability if a random sample is generated according to π_T, with T being very small. One difficulty with this approach is that when T is very small, the time is takes for Markov chain to reach equilibrium can be excessive. The SA algorithm tries to resolve this drawback by using a slow cooling rate $T(t)$.

The SA can be viewed as a local search algorithm in which that there are occasional upward moves that lead to cost increase.

Assume that $X_T(t)$ is irreducible and periodic. According to this assumption, SA algorithm converges if $\lim_{t\to\infty}(P[x(t) \in S^*]) = 1$. SA convergence condition according to Hajek is presented next.

Theorem (Hajek, 1988): state i communicates with S^* at height h if there exists a path in S, with each element of the path being neighbor of the preceding element. The path starts at i and ends at some element at S^* and such that the largest value of J along the path is $J(i) + h$. Let d^* be the smallest number such that every $i \in S$ communicates with S^* at height d^*. Then, the SA algorithm converges if and only if:

$$\lim_{t\to\infty}(T(t)) = 0 \qquad (4.3)$$

and,

$$\sum_{t=1}^{\infty} \exp\left[-\frac{d^*}{T(t)}\right] = \infty \qquad (4.4)$$

$$T(t) = \frac{d}{logt'} \qquad (4.5)$$

where d is a positive constant. Hajek theorem states that SA converges if and only if $d \geq d^*$.

The constant d^* is the measure of the difficulty of x(t) to escape the local minima and travel from a non-optimal state to S^*. A problem with $d^* > 0$, in the sense that the problem has at least one local minima which is not the optimal solution, is the primary concern. In order to have an acceptable grasp on Hajek theorem, consider a local minimum with depth d^*. The SA makes an infinite number of trials to escape from it, and the probability of success at each trial, as discussed earlier, is $\exp(-d^*/T(t))$. Therefore, according to equation (4.4), an infinite number of trial will guarantee a successful escape.

In order to get more intuition on the interpretation of Hajeks' theorem, the connection between SA and the Markov chain is further analyzed. Formally, the statistics of Markov chain $x(t)$ under a slowly variation cooling schedule T(t) remains fairly unchanged if the cooling schedule is used in which the temperature is held constant for a long time period. Let $t_k = 1$ and $t_{k+1} = \exp(kd)$. Then let $\hat{T}(t) = 1/k$, for $t_k \leq t \leq t_{k+1}$. Consider the kth element $[t_k, t_{k+1}]$ of the piecewise constant schedule $\hat{T}(t)$. In order to study the convergence of the chain $x_{1/k}(t)$, the eigenvalues of its transition probability matrix is real. Its relaxation time is determined by its second-largest eigenvalue λ_2 for which good estimates are available, at least in the limit as $k \longrightarrow \infty$. e.g., Chiang and Chow, 1988 and Holley and Stroock, 1988. In particular, if the cost function J has a unique global minimum, the relaxation time is approximated by $exp(kd^*)$, which is the same constant d^* defined in the Hajek theorem. This gives more solid evidence on the convergence condition $d > d^*$ for the schedule $\hat{T}(t)$. If $d < d^*$, then it means that at each temperature 1/k, $x_{1/k}(t)$ is run with a negligible fraction of its relaxation time which is not enough for $\pi_T(i;t)$ to stay close to $\pi_T(i)$. Whereas, if $d < d^*$, then the interval $[t_k, t_{k+1}]$ corresponds to $\exp(k(d^* - d)$ relaxation times of $x_{1/k}(t)$ which implies that $\pi_T(i;t)$ is very close to $\pi_T(i)$ as k tends to ∞.

In practice, despite the lack of solid theoretical justification of SA convergence speed, SA was widely used by researchers in the past decades. Generally, the performance of SA is mixed; in some cases, it outperformed the best known heuristics for these cases, and, in others, heuristics performed better. The choice of the cooling schedule influences significantly the convergence of the SA, and hence, the quality of the solution generated.

To sum up, SA is a generally applicable and easy-to-implement probabilistic approximation algorithm which is able to generate good solution for an optimization problem.

4.1.3. Inter-digitated matching style

The device matching placement with inter-digitated matching style is one dimensional common centroid array as shown Fig. 4.1. The two devices are marked as A and B. Therefore, the matching pattern is AB_BA or AB_AB. Each Inter-digitated group G_i contains S_i devices, placed according to the bounding length and width, L_B and W_B respectively, for the whole group in the pattern AB_AB. L_G denotes the sum of S_i horizontal weights and N_S denotes the number of segments per row. The inputs of the algorithm are devices to be matched and number of device fingers per segment N_{fS}.

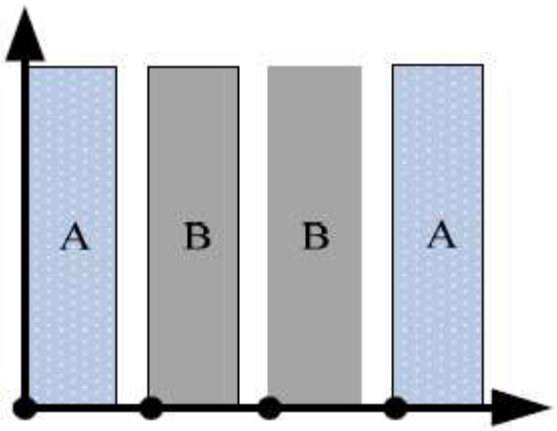


Figure 4.2 An example of inter-digitated array

Algorithm 4.1: interdig(G_i, N_{fS}, L_B, W_B)

1. // calculate coordinates of devices fingers placement.
2. // initialize m, RelX, RelY with 0
3. while (m < number of fingers per device) DO
4. FOR each device S_i DO
5. FOR each finger in segment range from 1 TO N_{fS} DO
6. find x-position PosX = RelX; // relative x position

7. find y-position PosY = RelY; // relative y position

8. increment RelX: RelX = RelX + Hweights;

9. y.max = max(y.max Vweights);

10. IF (RelX + Hweights > N_s*Hweights THEN {

11. RelX = 0;

12. RelY = y.max; }

13. m = m+N_{fS}

4.1.4. Common-centroid matching style

The matching of common centroid style requires centroids of matched devices to exactly coincide. Fig. 4.2 shows an example of matched devices by common centroid style.

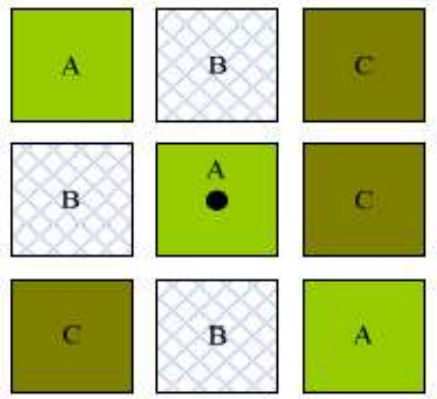


Figure 4.3 An example of common centroid array

Each common-centroid group G_i contains S_i devices, placed according to the bounding length and width, L_B and W_B respectively, for the whole group in which centroid of all devices should coincide. L_G denotes the sum of S_i horizontal weights, w_H and w_v denotes

finger horizontal and vertical weights respectively, N_f denotes number of device fingers, and N_S denotes the number of devices finger per row.

Algorithm 4.2: comcentroid(G_i, N_S, w_H, w_v)

1. // calculate coordinates of devices fingers placement
2. // initialize radprev, rad, Xrel, Yrel with 0
3. while (rad <=N_S*w_v) DO
4. increment rad: rad = rad + w_H;
5. Yrel = 0;
6. find x-position: Xpos = Xrel;
7. find mirror x-position: Xneg = -Xpos -w_H;
8. while (Yrel < N_f*S_i/(N_s ∗ 2)) DO {
9. find y-position: Ypos = Yrel;
10. find mirror y-position: Yneg=-Ypos-w_v;
11. P[F_num] = list(Xpos Ypos);
12. P[F_num+1] = list(Xneg Yneg);
13. P[F_num+2] = list(Xneg Ypos);
14. P[F_num+3] = list(Xpos Yneg);
15. F_num = F_num+4;
16. increment relative position: Yrel = Yrel +w_v;
17. increment relative position: Xrel = Xrel +w_H;

18. F[i] = F_num;

19. i = i+1;

20. // initialize k, s with 0

21. while (k < N_f) DO {

22. // find number of device fingers per row:

23. F_{Rnum} = F[s]/NUM(S_i);

24. FOR each device S_i DO

25. FOR each device finger m range from k TO min(F_{Rnum} N_f-k) DO

26. Posx.finger = nth(0 P[k+m]);

27. Posy.finger = nth(1 P[k+m]);

28. s = s+1;

29. k = k+F_{Rnum}*NUM(S_i); }

4.2. Optimization-Based Router

After placement, specific legal routing must be found for the wires needed to connect the circuits. The techniques typically applied to generate such routing are sequential in nature, treating one wire at a time with incomplete information about the positions and effects of the other wires. Annealing is inherently free of this sequence dependence. Nets with many pins must first be broken into connections-pairs of pins joined by a single continuous wire. This "ordering" of each net is highly dependent on the nature of the circuits being connected and the package technology

Based on simulated annealing algorithm [54], the router starts from the attained placement, after constructing routing channels to ensure the reliability and routability of the placement solution. The router requires modules terminal positions, allowed routing layers, and technology design rules to generate a DRC clean routing. The cost function which computes the probability of accepting a candidate net is given by:

$$P = min\left(1\, e^{-\frac{\Delta D}{D_{old}}\cdot\frac{1}{T}}\right) \qquad (4.6)$$

Where T is a constant-rate decaying temperature and ΔD presents the difference between the new and the old distance between the routed net and the destination terminal, in the sense that ΔD becomes more negative as the routed net approaches the destination. Distance between the candidate net and the target pin is calculated by;

$$D = min(\, abs(X_2 - X_1')\;\; abs(X_1 - X_2')\,) + \; min\big(abs(Y_2 - Y_1')\; abs(Y_1 - Y_2')\big) \qquad (4.7)$$

The probability P is then compared with a threshold constant r. A candidate net is accepted if $P \geq r$. Hence, chosen net is the one with the least cost, i.e., minimum wirelength.

During routing, each net is instantiated with its electrical constraints, e.g. current density, according to designer preferences, which are automatically converted to the corresponding wire width and layer according to a lookup table generated from the

technology file used. The algorithm searches for the minimum metal width satisfying the rms current density specified by the designer, according to available routing layers and the blockages surrounding the routed net within the DRC spacing specified for each blockage layer. The minimum DRC spacing allowed for each metal layer is defined by; the width, the layer of examined metals, and the length of the part in which metal lines are in a close proximity. Given a number of routing layers, each net is routed with a different metal layer in the presence of obstacles, e.g. wires, in order to ensure minimum wirelength. Metal lines are forbidden to pass over the devices. Multiple power straps are generated using reserved metal layers in Manhattan-like style to account for supply drop and hence prevent performance degradation.

5.Experimental Results

OASYN framework is implemented in 10,000 lines of code using SKILL programming language on a 2.4-GHz core i3 processor with 2GB of memory. Table 3,4, and 5 show simulated results of the circuit synthesizer for Folded Cascode OpAmp topology. Experiments are implemented using 65nm TSMC technology node. Table 6 shows simulated results of the circuit synthesizer for the same topology accounting for process, temperature, and supply variations with the minimum specs reported. Table 7 shows detailed simulated results for each corner.

Based on the MCNC benchmark circuits shown in Table I, experiments on area optimization, convergence speed, and convergence stability are conducted for each representation in the literature. Number of modules, I/O pads, nets and pins of the benchmark circuits are shown in Table I. Area and run time comparisons among different floorplan representations; SP, O-tree, B*-tree, enhanced O-tree, CBL, TCG, and TCG-S are shown in Table 2. TCG-S employing TCG-S* perturbing algorithm achieves almost the state-of-art area usage for the five benchmark circuits at the highest convergence speed.

Figure 5.1 shows the placements for the devices sizings indicated in Table 5 for simultaneous area and matching constraints optimization. Figure 5.2 shows the placement and routing results. Figure 5.3 shows the DRC Error messages of which there are no DRC spacing errors included (only density and CAD layer errors).

Table 2. Area and Runtime Comparisons among SP (On Sun Sparc Ultra60), O-Tree (On Sun Sparc Ultra60), B -TREE (On Sun Sparc Ultra 60), Enhanced O-Tree (On Sun Sparc Ultra60), CBL (On Sun Sparc 20), TCG (On Sun Sparc Ultra60), TCG-S (On Sun Sparc Ultra60), and TCG-S* (On Intel Core-i3) for Area Optimization

Circuit	SP		O-tree		B*-tree		Enhanced O-tree		CBL		TCG		TCG-S		TCG-S*	
	Area (mm^2)	Time (sec)	Area (mm^2)	Time (sec)	Area (mm^2)	Time (sec)	Area (mm^2)	Time (sec)	Area (mm^2)	Time (sec)	Area (mm^2)	Time (sec)	Area (mm^2)	Time (sec)	Area (mm^2)	Time (sec)
apte	48.12	13	47.1	38	46.92	7	46.92	11	NA	NA	46.92	1	46.92	1	46.92	0.2
xerox	20.69	15	20.1	118	19.83	25	20.21	38	20.96	30	19.83	18	19.796	5	20.74	0.62
hp	9.93	5	9.21	57	8.947	55	9.16	19	66.14	32	8.947	20	8.947	7	9.37	10

Table 3. Folded Cascode OpAmp Synthesis Results

Metric	Specifications	Simulated Results	Synthesized Circuit Parameters
Open Loop Gain (dB)	60	60	L1 = 228n. L3 = 490n. L5 = 500n.
GBW (HZ)	350M	398M	L7 = 3.6u. L9 = 2.7u. Lss = 510n.
Phase Margin (degree)	60	65.87	W1 = 221u. W3 = 51.8u. W5 = 21.4u.
Current Consumption (mA)	2	1.75	W7 = 305u. W9 = 5.3u. Wss = 1.58u
Output Swing (v)	0.8	0.9898	Vb1 = 0.642. Vb2 = 0.439
Slew Rate (v/us)	none	230	Mt9 = 46. Mtss = 54
Load Cap. (pF)	1		
VICM(v)	0.5		

Table 4. Folded Cascode OpAmp Synthesis Results

Metric	Specifications	Simulated Results	Synthesized Circuit Parameters
Open Loop Gain (dB)	60	60	L1 = 258n. L3 = 550n. L5 = 530n. L7 = 3.6u. L9 = 2.16u. Lss = 480n. W1 = 252u. W3 = 114u. W5 = 40.7u. W7 = 557u. W9 = 5.2u. Wss = 8.914u Vb1 = 0.642. Vb2 = 0.449 Mt9 = 82. Mtss = 80.
GBW (HZ)	600M	605.2M	
Phase Margin (degree)	55	58.19	
Current Consumption (mA)	3	2.88	
Output Swing (v)	0.8	1.006	
Slew Rate (v/us)	none	542	
Load Cap. (pF)	1		
VICM (v)	0.5		

Table 5. Folded Cascode OpAmp Synthesis Results

Metric	Specifications	Simulated Results	Synthesized Circuit Parameters
Open Loop Gain (dB)	60	60	L1 = 258n. L3 = 550n. L5 = 500n. L7 = 3.6u. L9 = 1.77u. Lss = 480n. W1 = 355u. W3 = 170u. W5 = 56.8u. W7 = 800u. W9 = 5.1u. Wss = 13u Vb1 = 0.642. Vb2 = 0.474 Mt9 = 100. Mtss = 117.
GBW (HZ)	0.8G	0.81G	
Phase Margin (degree)	50	51.66	
Current Consumption (mA)	4	3.797	
Output Swing (v)	0.9	1.055	
Slew Rate (v/us)	none	794	
Load Cap. (pF)	1		
VICM (v)	0.5		

Table 6. Folded Cascode OpAmp Synthesis Results on Process, Voltage, and Temperature Corners

Metric	Specifications	Simulated Results (min)	Post Layout Simulated Results (min)	Synthesized Circuit Parameters
Open Loop Gain (dB)	50	52.7	43.3	L1 = 200n. L3 = 520n. L5 = 500n.
GBW (HZ)	200M	251M	136M	L7 = 3.6u. L9 = 1.2u. Lss = 300n.
Phase Margin (degree)	50	56.8	52.4	W1 = 156u. W3 = 34u. W5 = 52u.
Current Consumption (mA)	2	1.51	1.09	W7 = 60u. W9 = 6u. Wss = 20u
Output Swing (v)	0.7	0.74	0.62	Vb1 = 0.642. Vb2 = 0.48
Slew Rate (v/us)	none	148.7	122.3	Mt9 = 10. Mtss = 28.
Load Cap. (pF)	1			
VICM (v)	0.5			

Table 7. Folded Cascode OpAmp Synthesis Results on Process, Voltage, and Temperature Corners

Corners	Process	SS				FF				SF				FS				TT			
	Temp	0		80		0		80		0		80		0		80		0		80	
	Supply	0.9	1.1	0.9	1.1	0.9	1.1	0.9	1.1	0.9	1.1	0.9	1.1	0.9	1.1	0.9	1.1	0.9	1.1	0.9	1.1
Metric	Gain(dB)	61.5	53.6	59.5	52.7	60.1	59.9	56.7	56.3	60.4	55.6	57.5	54.0	60.7	59.7	58.1	56.6	60.9	58.2	58.4	55.77
	GBW(MHz)	251	306	298	251	430	542	394	427	381	361	338	287	284	525	334	412	345	458	353	357
	PM(deg)	56.8	77.4	60.8	79.3	58.6	62.5	63.3	67.2	60.0	74.1	66.7	77.4	56.8	62.8	59.8	67.3	57.57	67.6	62.1	71.9
	I(mA)	0.40	1.33	0.69	1.4	0.84	1.44	1.1	1.5	0.76	1.4	1.03	1.5	0.45	1.4	0.76	1.43	0.60	1.4	0.90	1.46
	Swing(v)	1.24	0.99	1.02	0.78	0.97	1.0	0.74	0.77	1.11	1.0	0.87	0.78	1.11	1.02	0.88	0.80	1.11	1.02	0.89	0.80
	SLR(v/us)	148	553	274	582	336	590	445	617	303	586	417	612	172	561	304	589.6	236	575	362	601
	Load Cap.(pF)	1																			
	VICM (v)	0.5																			

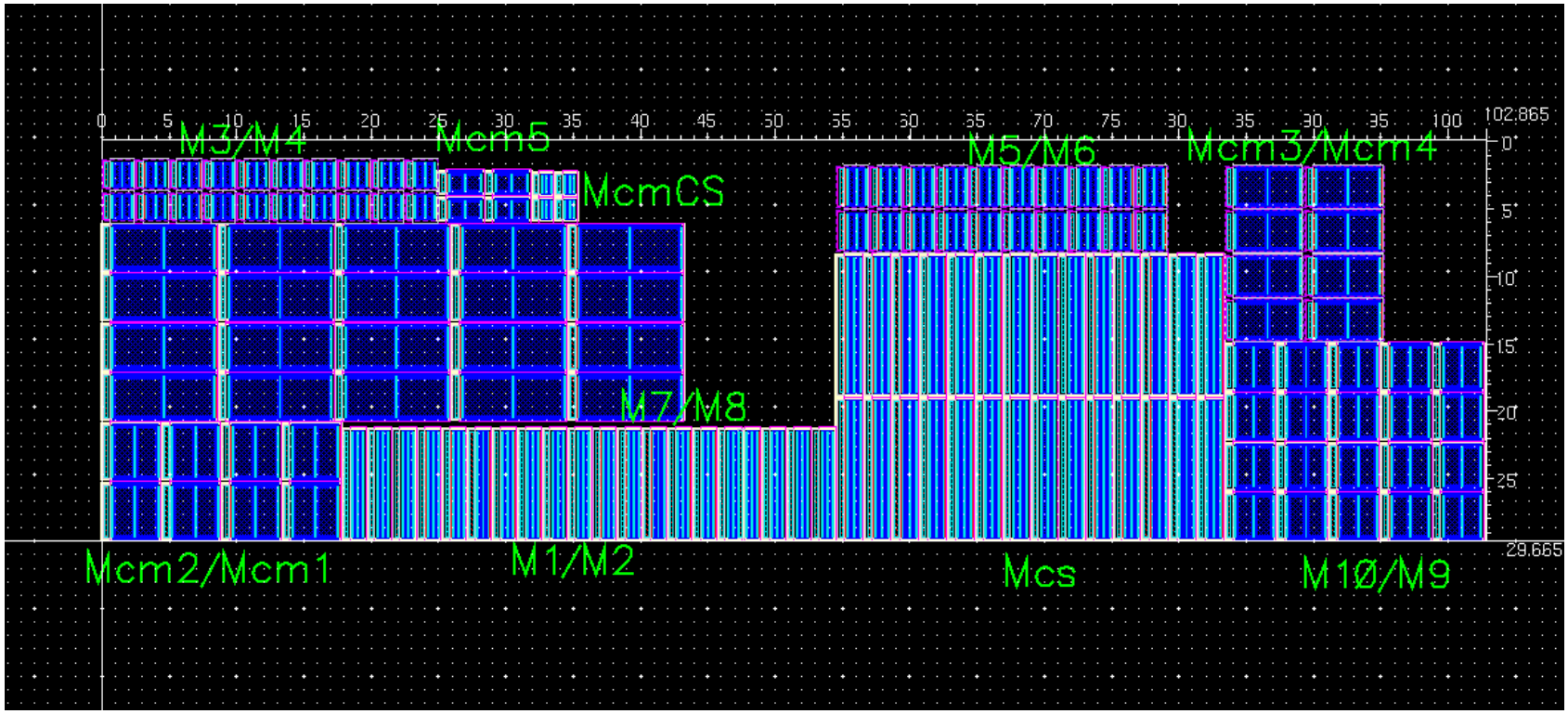


Figure 5.1. Generated Folded Cascode OpAmp Layout with the Common Feedback Circuit for Simultaneous Area and Matching Constraints Optimization. Area = 29.665x102.065 um2

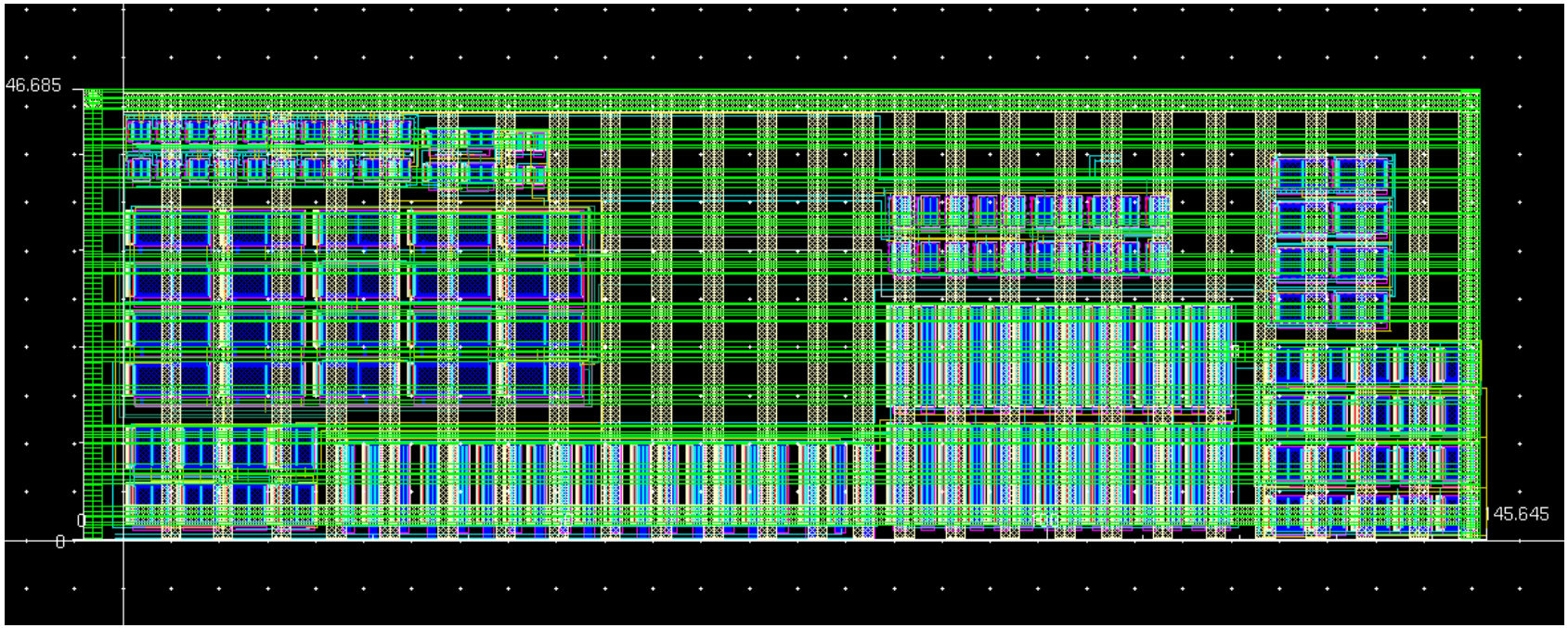


Figure 5.2 Automated Placement and routing solution (Area = 146*47 um2)

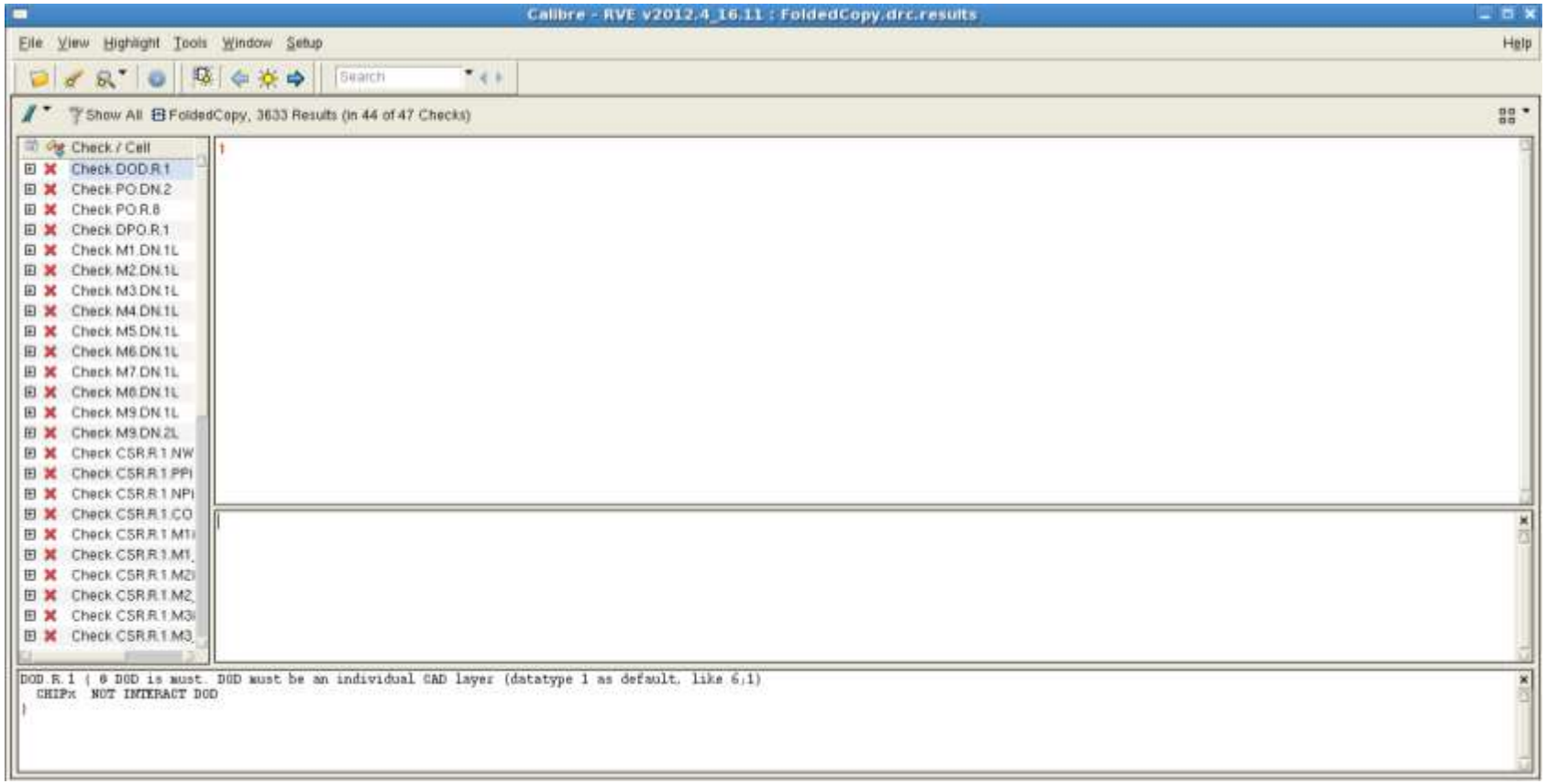


Figure 5.3 Calibre DRC Message of the placement solution

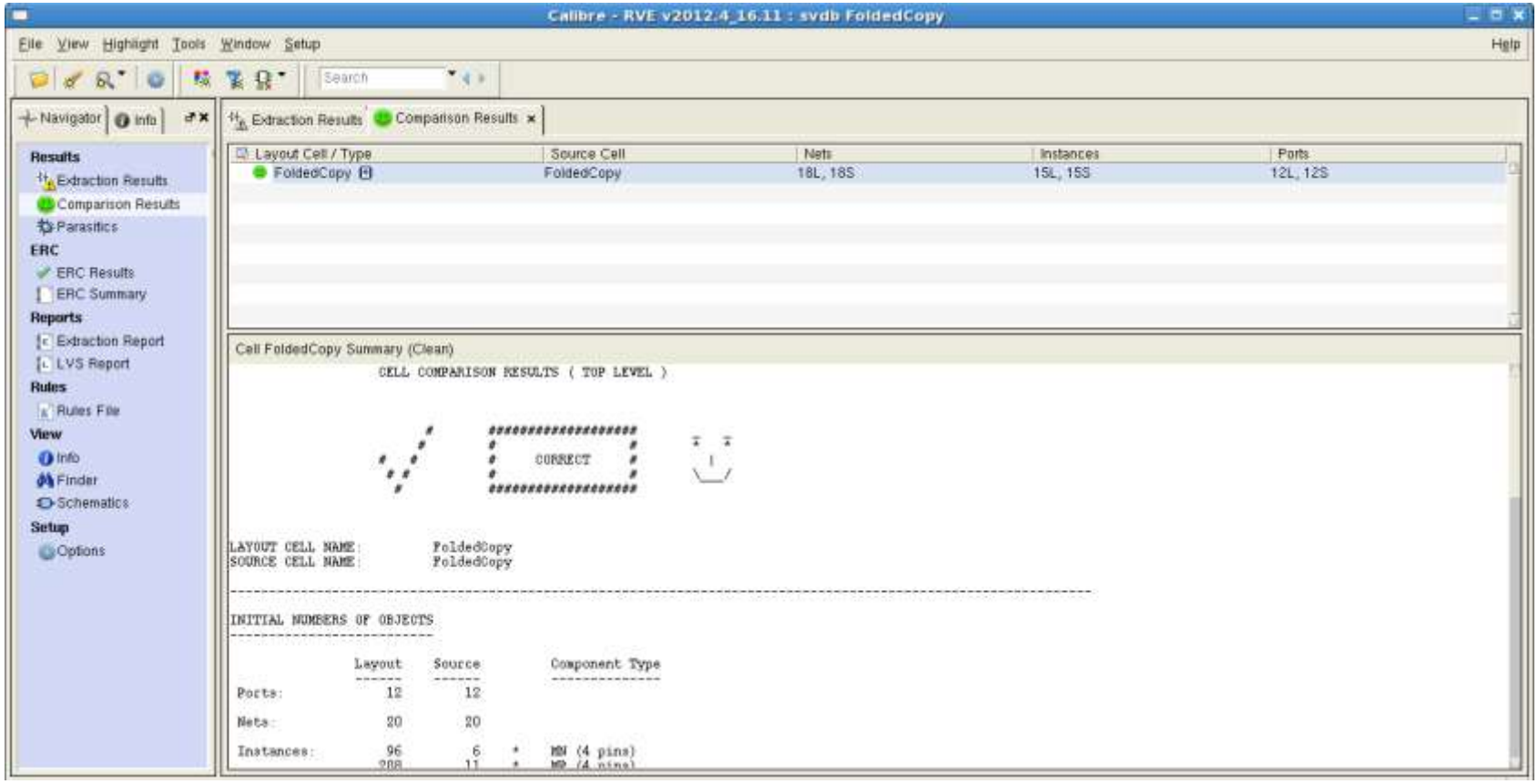


Figure 5.4 Calibre LVS Message of the layout solution

Conclusion

In this Thesis, a framework is presented for synthesis of operational amplifiers on the cell-level. The tool optimizes the design on both circuit and layout phases by exploring the corners design space and optimizing on worst case solution. Although the results shown are promising, yet other constraints and optimization factors need to be weighed into the tool design flow. The tool undermines the effects of boundary constraints, isolation constraints, and total wirelength of the routed nets. Floorplan area optimizer showed state-of-art results as optimization is applied on relatively few number of blocks. However, as number of blocks increase, the optimizer finds it more difficult to search for the optimum solution compared to other representations. Hence, a complexity analysis for TCG-S* based area optimizer is required to be studied. Considering the circuit synthesis tool, area optimization was only introduced in a later stage limiting the design space for area-power optimization. Applying the aforementioned enhancements and upgrading the tool on the system level can assist in the introduction of the concept of optimized standard-cell, which is well-established in the digital flow, in analog design.

Future Works

- Simultaneous optimization on area and wirelength. Wirelength of a net is estimated by half perimeter of the minimum bounding box enclosing the terminals of the net.
- Could SA be trapped in a local maxima?
  Simulated annealing can be applied to reduce the effect of the highly non-linear non-monotonic behavior of the model.
- Perform Sobol's sensitivity analysis on other amplifier topologies, e.g., Two-Stage Miller compensated OTA, to prove the universality of the algorithm and its minor dependency on the law and the model.
- Area Power optimization can be introduced earlier in the design stage, either by a rough calculation of the area based on the device gate dimensions or by looping through schematic and layout phases.

References

[1] M. Degrauwe et al., “IDAC: An interactive design tao1 for analog CMOS circuits,” IEEE J. Solid-State Circuits, vol. 22, pp. 1106-1 115, Dec. 1987

[2] R. Harjani, R. A. Rutenbar, and L. R. Carley, “OASYS: A framework for analog circuit synthesis,” IEEE Trans. Computer-Aided Design Integrated Circuits and Systems, vol. 8, no. 12, pp. 1247–1266, Dec. 1989.

[3] H. Y. Koh, C. H. Séquin, and P. R. Gray, “OPASYN: A compiler for CMOS operational amplifiers,” IEEE Trans. Compute.-Aided Design Integrated Circuits and Systems, vol. 9, no. 2, pp. 113–125, Feb. 1990.

[4] S.K. Gupta and M.M. Hasan, “KANSYS: a CAD tool for analog circuit synthesis,” in Proc. of Ninth International Conference on VLSI Design, pp. 333 – 334, 1996.

[5] N. Fujip, “Second Order Sensitivity Analysis for a Class of Shape Optimization Problems”, In Proc. IEEE 20th International Conference on Industrial Electronics, Control and Instrumentation, Sep. 1994, pp. 1176-1178.

[6] F. M. E1-Turky and R. A. Nordin, “BLADES: An expert system for analog circuit design,” in Proc. Int. Conf. Circuits Syst., 1986, pp. 552–555.

[7] H. Yang, A. Agarwal, and R. Vemuri, “Fast analog circuit synthesis using multi-parameter sensitivity analysis based on element-coefficient diagrams,” in Proc. IEEE Comput. Soc. Annu. Symp. VLSI, Tampa, FL 2005, pp. 71–76.

[8] H. Yang, R. Vemuri, “Efficient Temperature-Dependent Symbolic Sensitivity Analysis and Symbolic Performance Evaluation in Analog Circuit Synthesis”, In Proc. IEEE Design, Automation and Test in Europe, Mar. 2006, pp. 1-2.

[9] R. H. J. M. Otten, “Automatic floorplan design,” in Proc. Design Auto-mation Conf., 1982, pp. 261–267.

[10] D. F. Wong and C. L. Liu, “A new algorithm for floorplan design,” in Proc. Design Automation Conf., 1986, pp. 101–107.

[11] X. Tang, R. Tian, and D. Wong, “Fast evaluation of sequence pair in block placement by longest common subsequence computation,” IEEE Trans. CAD ICs., vol. 20, no. 12, pp. 1406–1413, Dec. 2001.

[12] J.-M. Lin and Y.-W. Chang, "TCG: A transitive closure graph-based representation for general floorplans," IEEE Trans. VLSI Syst., 2003.

[13] P.-N. Guo, C.-K. Cheng, and T. Yoshimura, "An O-tree representation of nonslicing floorplan and its applications," in Proc. Design Automation Conf., 1999, pp. 268–273.

[14] X. Hong, G. Huang, T. Cai, J. Gu, S. Dong, C.-K. Cheng, and J. Gu, "Corner block list: An effective and efficient topological representation of nonslicing floorplan," in Proc. Int. Conf. Computer-Aided Design, 2000, pp. 8–12.

[15] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence pair," IEEE Trans. Computer-Aided Design, vol. 15, pp. 1518–1524, Dec. 1996.

[16] S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitani, "Module placement on BSG-structure and IC layout applications," in Proc. Int. Conf. Computer-Aided Design, 1996, pp. 484–491.

[17] Y. C. Chang, Y. W. Chang, G. M. Wu, and S. W. Wu, "B -trees: A new representation for nonslicing floorplans," in *Proc. Design Automation Conf.*, 2000, pp. 458–463.

[18] J.-M. Lin and Y.-W. Chang, 'TCG-S: Orthogonal Coupling of P*-admissible Rep-representations for General Floorplans." IEEE Trans. Computer-Aided Design, Vol. 24. No. 6, June 2004.

[19] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, Introduction to Algo-rithms, 2nd ed. New York: MIT Press/McGraw-Hill, 2001.

[20] G. Giclcn and R.A. Rutcnbar, "Computcr-Aidcd Dcsign of Analog and Mixed-Signal Integrated Circuits." Proceedings of the IEEE, 88(12): 1825-1852. Dcc. 2000.

[21] M. zakaria, M. Madbouly, M. A. El-Nozahi,, and M. Dessouky,"Knowledge-Based Design Automation of Highly Non-Linear Circuits Using Simulation Correction." Proceedings of the 15th International Conference on Microelectronics, Dec. 2003, pp. 46-49.

[22] C. Toumazou, C. A. Makris, and C. M. Berrah, "ISAID: A methodology for automated analog IC design," in *Proc. Int. Symp. Circuits Syst.*, 1990, vol. 1, pp. 531–555.

[23] E. Berkcan, M. d'Abreu, and W. Laughton, "Analog compilation based on successive decompositions," in *Proc. Des. Autom. Conf.*, 1988, pp. 369–375.

[24] Z. Ning, A. J. Mouthaan, and H.Wallinga, "SEAS: A simulated evolution approach for analog circuit synthesis," in *Proc. Custom Integr. Circuits Conf.*, 1991, pp. 5.2-1–5.2-4.

[25] K. Swings, S. Donnay, and W. M. C. Sansen, "HECTOR: A hierarchical topology-construction program for analog circuits based on a declarative approach to circuit modeling," in *Proc. Custom Integr. Circuits Conf.*, 1991, pp. 5.3/1–5.3/4.

[26] B. A. A. Antao and A. J. Brodersen, "ARCHGEN: Automated synthesis of analog systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 3, no. 2, pp. 231–244, Jun. 1995.

[27] N. C. Horta and J. E. Franca, "Algorithm-driven synthesis of data conversion architectures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 10, no. 16, pp. 1116–1135, Oct. 1997.

[28] T. McConaghy, P. Palmers, M. Steyaert, and G. Gielen, "Variation aware structural synthesis of analog circuits via hierarchical building blocks and structural homotopy," IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., vol. 28, no. 9, pp. 1281–1294, Sep. 2009.

[29] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.

[30] G. S. Hornby, "ALPS: The age-layered population structure for reducing the problem of premature convergence," in *Proc. Conf. Genetic Evol. Comput.*, M. Keijzer, M. Cattolico, D. Arnold, V. Babovic, C. Blum, P. Bosman, M. V. Butz, C. CoelloCoello, D. Dasgupta, S. G. Ficici, J. Foster, A. Hernandez-Aguirre, G. Hornby, H. Lipson, P. McMinn, J. Moore, G. Raidl, F. Rothlauf, C. Ryan, and D. Thierens, Eds., 2006, vol. 1, pp. 815–822.

[31] R. Martins, N. Lourenço, S. Rodrigues, J. Guilherme, N. Horta, “AIDA: Automated Analog IC Design Flow from Circuit Level to Layout”, Proceedings of International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), Seville, Spain, Sep. 2012.

[32] M. Dessouky, M.-M. Louerat, and J. Porte, "Layout-oriented synthesis of high performance analog circuits," In Proceedings of Conference on Design, Automation and Test in Europe (DATE), pp. 53-57, 2000.

[33] H. Habal and H. Graeb, “Constraint-based layout-driven sizing of analog circuits,” IEEE Trans. Computer-Aided Design Integr. Circuits Syst., vol.30, no. 8, pp. 1089–1102, Aug. 2011.

[34] F. Balasa, K. Lampaert, “Symmetry within the sequence-pair representation in the context of placement for analog design,” *IEEE Trans. CAD of IC’s and Syst.*, vol. 19, no. 7, pp. 721-731, 2000.

[35] K. Krishnamoorthy, S. Maruvada, and F. Balasa, “Topological placement with multiple symmetry groups of devices for analog layout design,” in Proc. IEEE Int. Symp. Circuits Syst., May 2007, pp. 2032 2035.

[36] S. Dong, Z. Zhou, X. Hong, “A New Constraint-Driven Placement Approach for Analog Circuits”, In Proc. IEEE 8th International Conference on Solid-State and Integrated Circuit Technology, 2006, pp. 1763 – 1765.

[37] L. Xiao and E. Young, “Analog placement with common centroid and 1-D symmetry constraints,” in Proc. IEEE ASP-DAC, Jan. 2009, pp. 353–360.

[38] J. Lai, M.-S. Lin, T.-C. Wong, and L.-C. Wang, “Module placement with boundary constraints using the sequence-pair representation,” in Proc. IEEE Asia and South Pacific Design Automation Conf., 2001, pp. 515–520.

[39] A.B. Kahng S. Reda, “Wirelength Minimization for Min-Cut Placements via Placement Feedback”, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, Vol. 25, no. 7, pp. 1301-1312, July 2006.

[40] L. Xiao, E. F. Y. Young, X. He, and K. P. Pun, “Practical placement and routing techniques for analog circuit designs,” in Proc. IEEE/ACM Int. Conf. on Comput.-Aided Des., 2010, pp. 675–679.

[41] Cheng-Wu Lin, Chun-Po Huang, Soon-Jyh Chang, Jai-Ming Lin. Routing-aware Placement Algorithms for Modern Analog Integrated Circuits. Circuits and Systems (MWSCAS), 2011. IEEE 54th International Midwest Symposium on. Pages: 1-4, 2011.

[42] H. Ou, H.C. Chien, Y. Chang, "Simultaneous Analog Placement and Routing with Current Flow and Current Density Considerations", In Proc. IEEE Design Automation Conference (DAC), May 2013, pp. 1-6.

[43] W. Liu, C. Koh, and Y. Li, "Optimization of Placement Solutions for Routability", In Proc. IEEE Design Automation Conference (DAC), May 2013, pp. 1-9.

[44] H. Zhou, C. Sham, H. Yao, "Congestion-Oriented Approach in Placement for Analog and Mixed-Signal Circuits", In Proc. IEEE 5th Asia Symposium on Quality Electronic Design, 2013, pp. 97-102.

[45] L. Zhang and Y. Jiang, "Global-routing driven placement strategy in analog VLSI physical designs," in Proc. MWSCAS, 2005, pp. 1239–1242.

[46] H. Yang, R. Vemuri, "Efficient Symbolic Sensitivity based Parasitic-Inclusive Optimization in Layout Aware Analog Circuit Synthesis", In Proc. IEEE 20th International Conference on VLSI Design, 2007, Jan. 2007, 201-206.

[47] L. C. Severo, A. Girardi, "Parameter Variation and Sensitivity Analysis of a Two-Stage Miller Amplifier", In Proc. IEEE Argentine School of Micro-Nanoelectronics, Technology and Applications, Oct. 2010, pp. 78-81.

[48] Yiu-Cheong Tam, Evangline F.Y. Young, Chris Chu. Analog Placement with Symmetry and Other Placement Constraints. Computer- Aided Design. Pages: 349-354, 2006.

[49] Ender Yilmaz, Gunhan Dundar. Analog Layout Generator for CMOS Circuit. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions in, 28(1). Pages: 32-45, 2009.

[50] Y. Wu, X. Zhang, L. Chen, S. Fang, "Automatic Placement for Matched Devices of Analog Circuits", In Proc. IEEE Int. Conf. on Natural Computation, July 2013, pp. 1723-1727.

[51] Sobol IM. Sensitivity estimates for nonlinear mathematical models. Mathematical Modelling and Computational Experiments 1993;1(4): 407–14.

[52] Crestaux T, Le Maitre O, Martinez JM. Polynomial chaos expansion for sensitivity analysis. *Reliability Engineering and System Safety*, 2009; **94**: 1161–1172.

[53] L. Dawei, Q. Zhou, J. Bian, Y. Cai X. Hong, "Cell Shifting Aware of Wirelength and Overlap", In proc. IEEE Quality of Electronic Design, Mar. 2009, pp. 506-510.

[54] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp.671–680, May 13, 1983.

[55] S.N. Adya and I.L. Markov. Fixed-outline floorplanning: Enabling hierarchical design. IEEE Trans. on VLSI Systems, 11(6):1120–1135, December 2003.

[56] M. Kayal, S. Piguet, M. Declerq, and B. Hochet, "SALIM: A layout generation tool for analog ICs," in *Proc. IEEE Custom Integrated Circuits Conf.*, 1988, pp. 7.5.1–7.5.4.

[57] S. W. Mehranfar, "STAT: A schematic to artwork translator for custom analog cells," *Proc. 1990 IEEE Custom Integrated Circuits Conf.*, pp. 30.2.1–30.2.3, 1990.

[58] E. Malavasi, J. L. Ganley, and E. Charbon, "Quick placement with geometric constraints," in *Proc. IEEE Custom Integrated Circuits Conf.*, 1997, pp. 561–564.