# Automatic Synthesis of Analog Layout: A Survey

*Mark J. Rentmeesters*

VLSI CAD Laboratory
Dept. of Information and Computer Science
University of California, Irvine

*ABSTRACT*

A review of recent research in the automatic synthesis of physical geometry for analog integrated circuits is presented. On introduction, an explanation of the difficulties involved in analog layout as opposed to digital layout is covered. Review of the literature then follows. Emphasis is placed on the exposition of general methods for addressing problems specific to analog layout, with the details of specific systems only being given when they surve to illustrate these methods well. The conclusion discusses problems remaining and offers a prediction as to how technology will evolve to solve them. It is argued that although progress has been and will continue to be made in the automation of analog IC layout, due to fundamental differences in the nature of analog IC design as opposed to digital design, it should not be expected that the level of automation of the former will reach that of the latter any time soon.

August 2, 1990

# Table of Contents

# Automatic Synthesis of Analog Layout: A Survey

*Mark J. Rentmeesters*

VLSI CAD Laboratory
Dept. of Information and Computer Science
University of California, Irvine

## 1. Introduction

So many integrated circuits designed today are digital devices that *digital* and *integrated* circuitry are often considered synonymous. This simplification however, overlooks the fact that the same advances in microminiaturization that have made digital integrated circuits possible can also be applied to the design of electronic systems that process information represented in a continuous, or *analog* form as well. Indeed, until very recently the demand for monolithic solutions to analog circuit needs has been quite small in comparison to digital ones. This however is changing as applications for integrated circuits expand. The increasing economy of microminiaturization has made possible new applications for this technology, many of them involving analog circuitry. In particular, with competitive demands forcing full systems to be placed on an ever shrinking number of chips, in an inherently analog world, it is becoming increasingly necessary to include at least some analog interface circuitry on many primarily digital ICs. For this reason, demand for analog circuitry has recently begun to catch up with that for digital. Today, analog and mixed analog-digital IC design starts represent a substantial and growing percentage of all integrated circuits under development.[19]

In fact, so fast has been the growth in demand for analog integrated circuitry in recent years that somewhat of a vacuum now exists in the availability of computer aided design tools for such circuit development. This situation does not represent the first time the integrated circuit design community has found itself in such a predicament, however. A similar situation existed in the late seventies, when the advent of VLSI levels of circuit integration left the mostly manual methods of IC design then in practice incapable of fully exploiting the technology's capabilities, even for digital applications. Since then many automatic design tools have come to the aid of the digital IC designer. The most powerful of these tools provide design *synthesis,* that is, they will generate designs or features of a design automatically without the need for detailed human direction. Unfortunately, few of the synthesis tools developed for digital IC design to date have proved applicable to the somewhat different task of developing analog ICs. Nevertheless, automation of analog integrated circuit design was never forgotten. It has merely been overshadowed by developments in the digital world. As will be seen in what follows, useful tools are already at hand, and prospects for continuing additions to their number look quit good.

This paper will review the state of the art in the automatic synthesis of the physical geometry or *layout* for analog integrated circuits. The *layout,* process, also called *physical design,* is a very influential part of the design of an analog integrated circuit. Layout deals with the geometric definition, i.e. construction and positioning, of electrical devices and their interconnections on the surface of the an IC chip. Although there are now several excellent reviews of automatic layout synthesis in general,[36, 45, 53, 62, 71, 80] for the most part these works describe only digital layout. What is and is not applicable to analog design is not made clear in these expositions. It is hoped that the present work will provide this information.

Although an effort has been made to make the present work self contained, to keep the development concise and focused, emphasis will often be placed on the differences between analog design tasks and their corresponding digital counterpart. This not only saves the need to describe the digital method or problem in extreme detail, but also emphasizes the difficulties unique to analog design automation that make it distinct from that of digital. Being primarily a survey of the literature in automatic analog IC layout synthesis, rather than of the technology itself or of its applications, this paper will be focused in several other ways as well. It will, for one, be concerned for the most part with analog circuit applications and fabrication processes already in widespread

use. That is, no coverage will be given to the automatic synthesis of experimental or as yet to be fully developed circuit technologies (e.g. neural networks), for which no publications describing specialized automated design aids currently exist. Likewise, the most heavily emphasized fabrication process will be CMOS. This is because it is for mixed analog-digital circuits that interest in alleviating analog circuitry design bottlenecks through automation is strongest, and CMOS is currently the technology of choice for mixed analog-digital IC design. Fortunately, the general principles of most of the automation techniques that will be described are not highly dependent upon the type fabrication technology employed. In addition, essentially no coverage will be given to so called *bipolar array* or *masterslice* analog IC development methodologies, as physical design in these processes is usually made very simple by intention, often requiring little or no automation. The reader interested in masterslice design techniques can consult (81).

It will be seen that the primary aim of most techniques described in this report is to provide an automated solution to some problem or group of problems in the design of high performance, full custom analog integrated circuits. Because these solutions are for full custom applications, they enjoy every design freedom that would be available to the human designer. But because the designs are for high performance circuits, their use cannot generally represent a substantial compromise over the results of human effort either.

The paper is organized as follows. In section two an overview of contemporary analog IC design and layout practices is presented. The background material presented in this section will allow the non-analog IC design specialist to properly comprehend and appreciate the problems and challenges involved in analog IC design and its automation. The objective of section three, the main body of the review, is then to provide an unbiased and exhaustive summary of progress reported to date in the automation of analog IC layout synthesis. The ordering of subsections within section three arranges the problems in automatic analog layout synthesis roughly from the simpler to the more challenging. This organization is also roughly chronological, as the more demanding challenges have tended to resist attempts at automation longer. Unbiased exposition ends in the final section, number four, where a critical analysis of the strengths and weaknesses of current efforts in analog IC design automation is attempted. After identifying the apparent sources of these strengths and weaknesses, the chronological progression begun in section three is projected into the future in section four with predictions of likely developments yet to come in the field of analog IC design automation in the next several years.

## 2. Analog ICs and Their Design[27, 28]

### 2.1. Characteristics of analog circuits

The variety of analog circuits in widespread use today is actually quite small. Application of electronics to signal and information processing began with the development of electronic telephony and radio nearly a hundred years ago. These early applications utilized a continuous waveform data representation, but since the advent of digital data processing techniques in the fourties, which have generally provided a much simpler design path for most signal and information processing applications, the uses of analog circuitry have for the most part remained unchanged. Thus, most analog circuits designed today can be classified as some form of signal amplifier, modulator, or demodulator, the principle classes of electronic circuits in existence when radio and telephone where the most extensive and demanding applications of electronics.

Today, the "amplifiers" are commonly referred to as *linear* electronics, and include the ubiquitous operational amplifier, or *opamp,* and many varieties of frequency selective amplifiers usually referred to as *filters.* Modulators and demodulators, which are in much less demand, are generally lumped together as *nonlinear* circuitry and include such devices as voltage controlled oscillators (VCOs -- the frequency of oscillation is controlled by an input voltage), waveform reshaping circuits (e.g. rectifiers), and waveform generators (that produce periodic patterns having various shapes, e.g. sinusoidal, square). Not surprisingly, the only analog circuit types to have effectively arisen since the advent of digital technologies have been analog-to-digital (A/D) and digital-to-analog (D/A) converters. Something that all these circuit forms share in common, at least when compared to modern digital electronics, is simplicity of function. Their output has a very direct relationship to their inputs. This is because a system having a complicated relationship between its input and its output, especially if this relationship requires long term information storage, quickly becomes very difficult to design as an entirely analog circuit.

The difficulting of design arises from the higher level of electrical detail utilized in analog circuits than in digital ones. The greater importance of pre-

cision in voltage levels, current rates, and so on in analog circuits means the electrical properties of their components, such as resistances and capacitances, have to be determined and specified more precisely. It also means more attention has to be paid to less influential device characterists and other physical phenomena than is necessary in digital design. For example, electromagnetic coupling between unconnected but adjacent wires can often be safely ignored in digital electronics but not in analog. For this reason, when assessing the design complexities of integrated circuits, the often low component count of analog circuits as compared to digital can be deceptive. Although few in number, the components in an analog circuit must often be specified at a much higher level of detail than do those in digital circuits. For example, at the waveform level of detail employed in analog circuitry, seemingly "simple" circuit modifications, such as alteration of a device characteristic (e.g. resistance) by a few percent, can significantly alter circuit behavior. Thus, analog circuits tend to be simple in structure but complex and precise in their detail.

Nothing better conveys the subtlety of detail present in seemingly simple analog circuits better than their performance specifications. Analog circuits are notorious for their apparently inexhaustible measures for characterization. A single example, that of an operational amplifier, will serve to illustrate this well.

An ideal operational amplifier is a voltage controlled voltage source extracting no energy from the controlling input circuit (having infinite input impedance), providing arbitrarily high output power (having zero output impedance), always responding without delay to any change in input, as well as being impervious to any other changes in its environment. Some of the parameters commonly used to describe how well any physically realizable op amp approximates this ideal are the following:[29] *Gain,* a measure of the power produced on output for a given power on input; *Offset voltage,* the non-zero constant error voltage appearing on output when none should appear; *Bandwidth,* a measure of the range of input frequencies over which useful amplification can be expected -- generally, as input frequencies increase, amplification decreases; *Slew rate,* a measure of how fast the output responds to changes in input; *Output resistance,* a limiter of output power; *Dynamic range,* a measure of the minimum and maximum voltage levels between which the output can be expected to have a linear relation to the input; *Power supply rejection ratio,* a measure of the extend to which variation (noise) in power supply voltages will appear in the output.

It should be pointed out that for any real circuit, most of these specifications are dependent upon many environmental and circumstantial conditions, so that, for instance, a single characteristic, such as maximum gain, can be considered *at* any one of a number of different frequencies, operating temperatures, output loadings, and so on. With such possibilities for qualifying any performance characteristic, it can truly be said that there is no limit to the variety, number, or uniqueness of specifications that can define the objectives of an analog IC design task.

There are many signal processing applications in which the input and output are both analog, suggesting that a completely analog design solution would be the most natural. Nevertheless no such design implementations are ever attempted. Instead, a partly analog, partly digital solution is chosen. The choice of a mixed analog-digital design alternative in situations in which an entirely analog one would seem more natural is usually justified on the basis of the shortcomings of analog circuit technology. The shortcoming most often cited is insufficient signal representation accuracy. However, quite often these shortcomings, especially representation accuracy, are not fundamental to the technology itself, but only describe limitations for the *easily developed* design possibilities.

For example, for many analog signal processing applications, the only inherent limitations deriving from representation accuracy are those associated with the original input or the output. These limitations would be experienced by any partially digital solution as well.[84] A completely analog solution competitive with a partly digital one would therefore, in principle, be possible to design. However, designing the entirely analog solution would probably be woefully more difficult than designing a partly digital one. Designing the system as a completely analog circuit would require that its behavior be predictable and understandable at the continuous waveform level of detail through out. With a partly digital implementation, this need not be the case.

Thus, even though an analog circuit's function may be a simple one, this is typically the case because the complexity of the circuit's design has limited what could be accomplished using analog IC design methods alone.

## 2.2. Analog IC Design

### 2.2.1. Topology and component selection

A circuit's *topology* is a specification for the connectivity of its components, that is, it defines what is connected to what. It should not be confused with the more geometric *layout* topology it may acquire if implemented monolithically. Together with a specification of the electrical properties of the components themselves, it very often is the only complete, electrically abstract, constructive definition of an analog IC's intended design. It does not include the arrangement of the components or any physical characteristics of these components that do not have direct electrical significance. For an integrated circuit, these properties would be determined later by the circuit's layout. Conceptually at least, topology and component selection precede layout specification, although in practice, for analog IC design, the two are usually developed in close concert.

Analog circuits intended for monolithic manufacture are more constrained in their design than are circuits that can be constructed from large, discretely fabricated components for several reasons.

First, the smallness of economically viable IC circuits puts limits on the types and values of components that can be utilized. For example, at micron dimensions and audio frequencies, no significant inductances are achievable, hence there can be no inductors in circuits for these applications unless connections are made for them to be provided off chip. Similarly, higher values of resistance and capacitance are generally achieved through larger component size. Thus there is a practical upper limit on the values of resistors and capacitors employable before the economic advantages of microfabrication are exhausted.

The second reason monolithic analog design is more constrained is that analog circuits, as has been noted, are generally very sensitive to the precise values of their components, yet contemporary integrated circuit manufacturing technology does not provide the precision necessary to achieve good value rendition. Where as it is possible to obtain discrete components such as resistors or capacitors with manufacturing tolerances on there electrical properties of as little as one percent, the techniques used to provide many electrical characteristics on an integrated circuit commonly provide tolerances of only about 20 - 30 percent, and sometimes worse.

The third difficulty with monolithic analog IC design is achieving sufficient component and signal isolation. Although often so assumed in digital IC design, neither the silicon substrate nor even the vacuum above its surface is in reality an ideal insulator. For analog circuitry, components placed proximally often must be treated as electrically coupled, and since every component is of necessity close to the substrate, by extension, every component in a monolithic circuit is to some degree coupled to every other. This often forces the use of circuit topologies (e.g. "double ended" or "fully differential") that are relatively insensitive to some types of coupling. More will be said on component placement and coupling shortly.

Fortunately, because all components on an IC are manufactured simultaneously under identical conditions, monolithic manufacture of electrical components does have one redeeming virtue over discrete manufacture. Although the exact electrical parameters of individual, monolithically manufactured components may vary widely, all parameters of like kind on the same chip tend vary in concert, tracking one another closely. Thus *ratios* of parameters for devices manufactured monolithically, such as the ratio of capacitances of two capacitors, resistances of two resistors, or gains of two transistors, tend to *exceed* those that can be easily achieved with discrete components.

Exploiting this advantage while ameliorating the disadvantage of poor absolute tolerances means utilizing circuit topologies that rely less on absolute component values and more on component matching too achieve good performance. This, in fact, is a design practice that is now basic to all analog IC engineering.[18] Monolithic analog circuit topologies are generally chosen so as to be insensitive to individual component values, so long as these values properly *track* those of other components. Conversely, the ability to achieve precision component ratios is often what is exploited in monolithic analog IC design to achieve precision of operation. Thus, in monolithic analog circuit design, component *matching* plays a fundamental a role. It influences circuit topology selection and is often the principle limiter on many circuit performance specifications.

Since difficulty of understanding and predicting electrical circuit behavior at a continuous waveform level of detail is the limiting factor on analog circuit function complexity, it is also the principle challenge in analog circuit design. Unlike digital design, where boolean algebra and other simplifying abstractions can be applied with fairly high confidence to logical representations of circuit behavior, in analog circuit design, the models employed for circuit analysis are much more approximate and less encompassing. To

predict or explain the many and varied characteristics of interest in analog circuit design, it is often necessary to make simplifying assumptions that restrict the validity of the conclusions reached. Different models, employing different simplifying assumptions, and thereby having different ranges of validity, are used to predict or explain different characteristics of a circuit's behavior. Because of the limited applicability and reliability of circuit models for analog IC design, this design process is a much more experimental undertaking than is that for digital design.

Much more so in analog design than in digital, the only reliable arbiter of what a circuit will do is the actual circuit itself. Designers of discrete analog circuits rely very heavily on iterative experimentation through breadboarding to develop workable circuits. Due to the high cost and long delay involved in integrated circuit fabrication, monolithic analog designers must rely more on precise numerical simulation, but the methodology is nevertheless very much the same. A tentative design is put together and then tested. performance measures are taken, the design altered, and the cycle repeated until a suitable solution is found. In the end, what design models existed to systemize the search served only as guides, and the reasons for the circuit's exact, final behavior might have never been fully explained or understood.

The precision simulations employed in this search process are usually called *SPICE* or *SPICE-level* simulations after a commonly used circuit simulation tool. They are generally very computationally intensive. Thus this sort of cut-and-try development can be very time consuming, even for very small circuits, with only "a few" circuit parameters that must be determined experimentally. Note that this fairly explains the current de facto limitation on analog circuit complexity.

Besides being time consuming, it is also apparent that circuits designed in this way tend to be very fabrication process dependent. Of necessity, such empirical methodologies rely heavily on as many assumptions that can be identified and made as possible. This reduces the amount of search required and increases the likelihood that what is being observed in design is actually how the circuit will behave when manufactured. That is, an analog circuit is most often designed for a specific target fabrication process. Moreover, that one target process is generally characterized in much greater detail than it would be for designing digital circuits. The net result is a circuit that will perform properly when manufactured on the target fabrication process, and very likely *only* on that process.

Although the iterative experimentation often continues into physical design and through fabrication as well, at least initially, the result of all these considerations an of all this effort is a specification for the circuit's topology and for each of the devices it contains. Usually the devices have a principle characteristic, for example, resistance for resistors, which is called their *size*. Transistors usually have two characterists, length and width, also referred to as sizes. Thus component selection is often referred to as *device sizing*. Similarly, a circuit topology together with component descriptions is often referred to as a *sized schematic,* even though this phrase more traditionally has referred to a graphical representation (i.e. a drawing) of a circuit topology that indicates device sizes.

### 2.2.2. Analog IC layout[6, 29]

The actual physical form of an integrated circuit is determined by its layout. The layout is a specification of the location, geometric construction, and physical composition (e.g. metal or silicon) of all the components of a circuit and of their interconnections. No more design detail is usually necessary given this geometric definition in order to manufacture the circuit save to indicate which fabrication facility to use. With the addition to the circuit specification of the layout level of design detail, the idealizations of any higher level design abstractions come face to face with the actual characteristics of the physical world. The goals of layout definition are to render the ideal intended behavior of a circuit with sufficient fidelity, and to do so as inexpensively as possible.

For layout purposes, a circuit's ideal behavior is usually defined by a sized schematic, while its cost is typically taken to be its total layout area. Unlike layout of digital circuits, the challenge in physical definition of analog ICs is typically not achieving minimum area, but in keeping the implementation sufficiently close in behavior to that of the specification. Most of the difficulties involved in maintaining adherence to specification can be classified as one of two types, those due to intrinsic discrepancies between the physical and ideal known as *parasitics,* and those due to uncontrollable operational and process *variations.*

The terms "parasitics" and "parasitic components" refer to any nonidealities of an electronic circuit's physical implementation that can be modeled as additional components inserted into the intended circuit topology to better approximate

actual behavior. Such additional components alter the circuit's behavior, usually adversely, and therefore they generally must be minimized. The parasitics of primary concern in layout are those due to interconnections or *routing.* This is because individual device parasitics tend to be more predictable and therefore their effects are usually addressed prior to layout. Whether due to routing or other layout features however, the parasitics influenced by layout are generally of two types, resistive or capacitive.

The nonzero resistance of some routing layers, such as polysilicon and diffusion, can make them unacceptable for use as interconnections in critical areas, even over short distances. For example, a short polysilicon bridge or jumper joining to metal wires separated by a third can hurt performance in some situations. Likewise the resistance of interlayer contacts can sometimes be of concern. Thus in analog layout, routing cannot be considered a matter of just providing the proper connectivity in as little space as possible, as it must also concern itself with the effects of inserting numerous additional resistive electrical "devices" in the form of contacts and poly bridges. Fortunately however, for the most part, resistive parasitics can be adequately avoided by routing primarily in metalization layers, for which resistive parasitics are often negligible.

Of much greater concern are capacitive parasitics, as they are more pervasive, harder to avoid, and frequently more influential, due to their inducement of internodal coupling. Circuit specifications in the form of a sized schematic often assume that each electrical equipotential, i.e. *node* or *net,* is electrically isolated from every other (except where indicated by desired component connections), yet in any physical circuit realization, such is not the case. Any two electrical equipotentials in close proximity to each other constitute a capacitor capable of transmitting a time varying electrical signal. The process of translating a sized schematic into a physical layout thus introduces numerous extraneous capacitors that unintentionally couple nets to each other even when they are not directly connected. Unfortunately, at the signal energies, circuit sensitivities, and spacing distances of contemporary analog ICs, the coupling induced by these parasitic capacitances can adversely effect circuit performance, and therefore cannot be neglected.

Controlling the undesirable couplings caused by these extraneous capacitors, which result in "crosstalk" between and "noise injection" within nets carrying time varying signals, is a major challenge in analog IC layout. Unacceptable coupling between nets can result from wires that cross over one another or that run next to each other for an extended length, as in a routing channel. Signal carrying nets can also be indirectly coupled to each other through non-signal carrying nets. For example, two nodes, each coupled to the same power line by crossovers, can pass interference to each other through this intermediary shunt. The most notorious such intermediary shunt is that of the substrate. Due to the planar nature of IC layout, every electrical IC net is capacitively coupled to the region of the substrate over which it is laid out. The substrate, moreover, is usually a lightly doped semiconductor, not an insulator, and can thus pass signals, albeit with some attenuation, from any net to any other.

To obtain adequate levels of internodal isolation within analog ICs, crossovers of sensitive signal carrying nets must be avoided. Very often sensitive signal carrying nets must be "shielded" from other nets by adequate spacing that is much greater than the minimum possible, and/or by interposition of other less electrically active nets or material. The cumulative layout area of wiring that implements net connections must be monitored as this determines the extent of coupling to the substrate. Routing of far ranging, or *extensive* nets, such as power lines, must be carefully managed to avoid indirect coupling.

As will be seen in section 3, although not every net must be treated with such extreme care, in most analog IC applications, there are enough nets of such importance throughout the design that the overall layout organization and many layout details are determined by the desire to achieve adequate nodal isolation.

Just as influential, if not more so, in determining layout organization, is control of variations in electrical characteristics due to fluctuations in the manufacturing process and in circuit operation. As has already been described, the most important of these, especially at the layout design stage, is ensuring proper component matching. By the time an analog IC design is ready to be laid out, there are many devices whose matching properties critically influence circuit performance. In fact, circuit topologies for analog ICs are often chosen so that improved performance can be achieved if component matching is extended to parasitics as well. Thus whole subcircuits consisting of many components, interconnections and all their related parasitics may need to match those of other subcircuits for optimal overall circuit behavior.

Good matching, however, depends upon proper physical definition of the circuitry involved. Ulti-

mately, matching is always the result of uniformity in processing. Even within a single chip however, processing variations do exist from point to point. These variations are influenced by the kinds and types of devices being fabricated, and even by the local "environment" of structures close by. Because many of the causes of these variations are poorly understood, for the purpose of analog IC layout, controlling their effects pragmatically reduces, for the most part, to providing as much "sameness," or uniformity in geometric structure as possible. That is, components whose electrical properties should match are given similar if not identical geometric definitions, the same orientation, and placed as close to each other as possible, so as to share as nearly "the same" location as possible, and so on. The need to match parasitics and whole subcircuits results in a desire to achieve sameness in routing and intercomponent spacing as well, so that good analog IC layouts often have a remarkable degree of symmetry, and achieving such symmetry becomes a primary goal of the analog IC layout process.

Very often achieving proper matching is so valuable that it overrides the desire for minimum area or reduced parasitics. For example, extraneous parasitics, such as unnecessary crossovers or perfunctory routing, may be added to some parts of a circuit in order to achieve better matching with other parts where similar parasitics already exist and cannot be removed. It should be noted that although providing proper layout symmetry for matching purposes is usually not hard to achieve in practice, the layouts that display such symmetry are usually markedly different from those that would have resulted had no matching requirements been enforced. In particular, digital layout methodologies that do not consider matching are generally inappropriate for analog layout for this reason.

In addition to tolerances on matching, it is also necessary in laying out an analog circuit to consider variations in the actual values of each device. Although as has been mentioned, absolute tolerancing of monolithic device electrical characteristics is generally so poor that circuit topologies are usually selected to be relatively insensitive to them, no design is ever completely free from absolute device parameter values. Fortunately, absolute device tolerancing primarily involves devices individually, so that techniques for addressing this problem need only concern the relatively simple geometries of individual devices. For example, monolithic capacitors are generally laid out as square rather that oblong rectangles to minimize variations due to edge definition irregularities. Similarly, mask alignment errors can also cause deviations in device characteristics that can be partially compensated by giving the device a rotationally symmetric layout that is thus insensitive to the direction of mask offset.

Variations in device characteristics due to temperature is an exception to the need to consider absolute device tolerances only individually. Temperature can very from point to point and from time to time across a chip as determined by the location and functioning of various heat producing components. Proper control of device characteristics that vary with temperature thus requires some consideration when laying out a circuit as a whole. Heat producing components, for example, should be placed away from others adversely affected by heat. Sometimes feedback controlled substrate "heaters" must be added to a layout to maintain nearby device temperatures at constant, predictable levels. Device matching is also influenced by temperature, so that thermal gradients often must be included in the symmetrical design of a circuit. Heat related concerns are less frequently mentioned in literature describing CMOS applications, presumably because CMOS circuits are less power hungry, and hence less heat producing, than bipolar circuits, for which concern about heat is more common.

Although process variations and device-to-substrate parasitics can often be predicted prior to layout, routing and other inter-device parasitics can only be accurately ascertained once a layout has been defined. These, however, as has been noted, can be principle determiners of overall circuit performance, particularly for measures concerning noise. The importance of routing and inter-device parasitics, together with the need for good matching, is what makes layout so influential in the design of analog integrated circuits. Very often topology and component selection must go hand in hand with layout planning. For example, good analog IC designers may consider the actual physical requirements for crossovers or poly bridges when selecting a circuit topology. Or a circuit topology might be selected, and its layout begun when incompatibilities are identified between what can be accomplished physically and what was assumed, prompting modification of the original selection. The latter might occur, for example, after a detailed parasitic extraction of the layout revealed overly optimistic assessments of these quantities. The fact that physical design so extensively influences the overall design of analog circuits and of the design process as a whole often makes it a challenging job even for human designers to say nothing of its automation.

## 3. Automatic Analog Layout Synthesis

The many concerns that must be dealt with in producing a complete layout for an analog or analog-digital IC can be conceptually subdivided into on of five categories. Each category consists of interrelated problems, issues and tasks that are usually considered together or in close concert when a layout is developed. These categories are: chip or system level planning, module assembly, module generation, device generation, and design optimization.

System level planning encompasses problems having to do with the layout of the entire system as a whole, such as overall floorplanning. Although there are many difficulties involved in system level layout of analog ICs for which automation could be applied (see for example (64)), so little has yet been done in the way of automation on the problems in this area that it will prove most convenient to mention them only in passing. Most of the work in design automation concerning system wide issues has been done in conjunction with the more urgent needs of module assembly.

Module assembly and module generation both refer to the integration of geometric layout sections, or *cells,* into larger ones, but differ in the level of complexity of the subcells being combined. Module assembly deals with the arrangement and interconnection of relatively large, complete cells, usually into the floorplan of the system as a whole, while module generation has to do with the definition of the internal geometry of the more complex cells that are "assembled." As will be more clearly described in section 3.1, the distinction between these two types of component integration derives from a change in concerns as the level of complexity of the cells involved increases. As will be seen, module generation is generally more difficult than module assembly. By far, the greatest concentration of research and development in automatic analog IC layout synthesis has been and will most likely continue to be in module assembly and module generation, as these are the most difficult and time consuming jobs in analog layout synthesis. Accordingly, most of this survey will be devoted to work in these areas.

Device definition refers to the lowest level of electrically significant geometric organization, namely, the graphic design of the fundamental electrical components, such as transistors and capacitors. Like system level layout, device definition is most conveniently covered only in passing. Although proper layout of many varied and often circuit-specific devices found in analog ICs is as critical to correct layout synthesis as is any other part of the design process, its automation is relatively simple, and is usually performed in conjunction with module generation. Hence an adequate review of the state of the art in automated device definition will result from proper coverage of module generation.

The category of design optimization is somewhat of a catch all. Within it we mean to include not only optimization of the layout by itself, but also that of the design as a whole. It encompasses all activities concerned with managing the operation of separately conducted design tasks so as to produce a design of as high a quality as possible. It therefore includes any type of system-wide design task coordination as well. Automatic optimization of analog IC layout is a relatively new and apparently growing field of development. Although young, due to its obvious value and probable greater extent in the future, an attempt will be made to ascertain the state of the art in this area as well.

Many automatic synthesis tools for analog IC layout address problems within several of these categories simultaneously. Although the issues addressed by any one tool may vary, the individual problems they solve do not. It will therefore prove most expedient to review what has been done in automatic analog IC layout synthesis on a category-by-category basis, rather than by individual tools that have so far been developed.

### 3.1. Module assembly and floorplanning

Though the diversity of concerns in analog IC layout may be interrelated and complex, they do at least tend to be manageable locally. For instance, the number of components in a collection that must mutually match each other is usually small, and the components themselves usually common to a small subcircuit that can be laid out as a unit. Similarly, many critically sensitive nets connect only a few components, so that the routing of these nets need not be extensive nor difficult to isolate. Consequently; as larger and more complete subdivisions of an analog IC are integrated, the number and criticality of problems tends to decline. There are fewer and less critical sensitive nets to route, fewer matching requirements, and so on. Thermal gradients tend to be more uniform and hence less significant as module dimensions increase as well.

The level of integration at which matching demands have all been dealt with within the modules themselves is particularly significant. Once these demands have been met, the needs for layout symmetry are gone and hence module placement much

less constrained. Also, modules already consisting of several matching components or subcircuits tend to be more uniform in size, or at least more conformable to various uniform heights or widths. The most common modules of this kind are complete op amps and banks of matching devices like capacitors or resistors. Large individual capacitors or resistors that need not necessarily match anything can also be included in this category due to their lack of matching needs and similar size. The task of integrating analog modules having no mutual matching requirements will be called *module assembly.*

The predominant concern in module assembly is isolation, in all its forms: crosstalk between signal nets, indirect feed-through via power lines, and noise injection from the substrate. Capacitive and resistive parasitics can still be an issue at this integration level as well. The relative simplicity of module assembly makes its automation simpler, permitting, in particular, more general and comprehensive methods to be developed and applied. It also makes the task more like that of digital layout. In fact, as will be seen, most of the tools for automatic analog module assembly developed to date use methods originally devised for digital layout with little and in some cases no modification.

When more than ten or twenty modules are to be assembled, it is usually worth while to first impose some type of structure on their layout as a whole. For layouts involving tens of cells, the most common if not universal structure imposed is that of a "strip architecture." In a strip layout architecture, a common dimension, i.e. length or width, is imposed on the layout of each module (possibly in groups), so that they may be efficiently organized side-by-side into strips, and the strips then stacked into levels with space, "routing channels" being left between the levels for inter-cell connection (see figure 1). The strip architecture is indeed universal in standard cell design practices and there are many tools, primarily developed for standard cell layout, for automating the layout process given this architecture. Although use of fixed libraries of standard cells has been much less successful in analog design as it has in digital, the strip architecture itself is just as pervasive in analog layout as it is in digital.

This naturally has lead to efforts to employ standard cell automation tools and automation methodologies to analog layout. To obtain useful results however, some modifications to these tools have usually been necessary. With digital standard cell module assembly methods, placement of cells typically precedes all routing, with cell arrangement

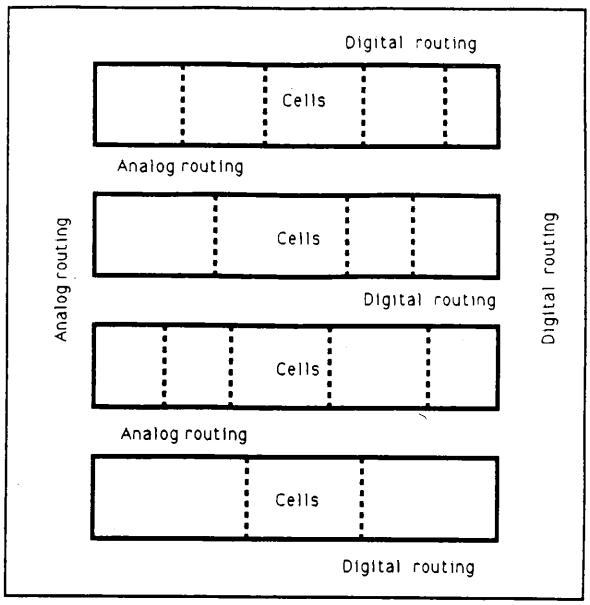


**Figure 1:** A strip architecture for analog layout

usually chosen so as to minimize the total length of all interconnect wiring. During placement, cells typically may be flipped, rotated and arranged without regard for net isolation problems that would arise in analog circuitry when the cells were subsequently interconnected. To simplify the task of routing under constraints imposed by crosstalk concerns, for analog layout, two adaptations of digital standard cell floorplanning and placement methods have been tried.

First, the strip architecture has been augmented with the requirement that every other routing channel between strips of cells be exclusively devoted to routing "sensitive" nets, as shown in figure 1.[12, 41] Thus each module has access to a channel dedicated to routing sensitive nets on one of its non-abutting sides, and access to a less exclusionary channel on its other side for noiser nets that should be separated from the sensitive ones. This channel segregation scheme is commonly used in manual layout of analog cells. It imposes an additional requirement on the layout of each module in that all external connections to sensitive nets must be made on one and the same side, and all connections to particularly noisy nets must be accessible from the other side. It does not however, restrain the overall cell assembly problem, as all cell arrangements are still routable. Within this structure, extensive power lines are usually routed as noisy nets in the "insensitive" routing channels, or along the edges of either channel as an integral part of the cells themselves.

Net isolation via an alternating channels separation scheme is usually insufficient by itself to guarantee workable analog layout when the modules themselves are only just large enough to address matching concerns as defined above. Current routing algorithms employed within either type of channel are as yet adaptations of relatively simple digital techniques, and there are usually still many inter-module sensitive nets with parasitic and isolation constraints that these routers cannot properly handle. To retain the ability to use simpler (and faster) routing tools where they can be employed, while still ensuring proper treatment of nets that these tools cannot handle, the alternating-channel adaptation is often accompanied by a second adaptation to conventional digital standard cell placement methods. This adaptation is to perform cell placement in an hierarchical fashion. Recall that the more sensitive nets tend to be local to specific subcircuits. Subcircuits containing nets with tight parasitic or isolation tolerances can thus often be clustered together into "polycells" or "macromodules" that can then be placed and routed as a whole. The critical nets internalized by this clustering process need never be seen by the routing or placement mechanisms determining the location and external connections of the cluster as a whole. Thus the higher level placement and routing algorithms can remain relatively simple, while the more arduous inter-cluster placement and routing has been localized and can be accomplished through other means.

When a hierarchical layout strategy is employed, many possibilities for inter-cell placement and routing, as well as for cluster generation itself, are possible. Inter-cluster layout can be performed manually (creating new cells in an analog "standard" cell library), by some modification of a conventional, general automation technique, or by automatic routines specialized for each type of cluster.

As an example of a more general automatic technique, Kimble et al.[42] constrain each cluster to occupy one strip, retain the global cell placement algorithm for inter-cluster cell ordering, and provide additional routing channels within the analog cells themselves for critically sensitive routing. (How inter-cluster routing is accomplished is not described.) They identify the clusters themselves through an analysis of the circuit's net list specification, which hierarchically describes the circuit in terms of subcircuits of various kinds, both analog and digital.

As another example, Winner et al.[85] attempt a completely automated method of both cluster generation and inter-cell placement. They group cells based on an analysis of net connectivities. Sensitive "summing" nodes are identified in the circuit's schematic and a tree of nets and components emanating from each of these nodes traced out in the net list until all remaining free terminals of each tree connect to insensitive nets. Each tree then constitutes a cluster. A cluster is then laid out in a single strip and routed within a routing channel internal to and dedicated for the cluster. This routing is accomplished by a special purpose algorithm.

As for the use of algorithms specialized for each type of cluster, note that this constitutes module *generation*. Hence, the hierarchical, poly-cell approach to cell assembly is as much an adaptation of conventional digital place and route methodologies to analog layout where they might not otherwise be applicable as it is an extension of these techniques into the realm of module generation, the topic of section 3.2.

Figure 2 shows an elaboration of the strip architecture of figure 1 that is very commonly used in both manual and automatic layout of switched capacitor circuits.[7-9, 69, 70, 87] Switched capacitor analog circuitry, especially switched capacitor filters, are unusually regular in construction, utilizing somewhat predictable ratios of three types of modules: op amps, banks of matching capacitors, and clock-controlled analog signal switches. This circuit regularity can be exploited to simplify layout with an overall floorplan that is more structured. With the scheme shown in figure 2, a dedicated routing channel for extensive but sensitive nets is placed below a strip exclusively containing op amps cells. This channel corresponds to the "sensitive" routing channel of the more general strip architecture. A routing channel for noisy digital nets corresponding to the earlier "insensitive," channel appears above a strip containing only switches controlled by clocks and other noisy digital nets. Between these two "global" routing channels are two "local" routing channels, one on either side of a median strip of capacitor cells. There are thus three types of strips: those for opamps, those for capacitor banks, and those for switches. Layouts consisting of multiple replications of this three-strip-type sequence are built up by alternating the order of the strips, so that outer digital routing channels and outer sensitive routing channels always coincide with their like kind, just as in the general strip architecture.

For this floorplan structure, again, clustering is generally employed and can be derived, just as before, from the circuit's hierarchical definition or from its connectivity. Both of these methods are

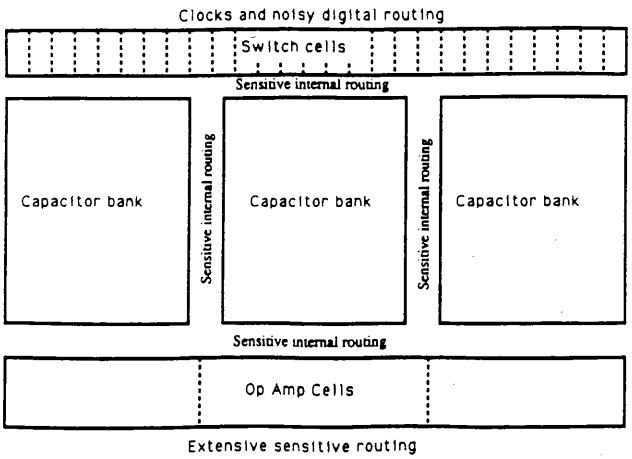


**Figure 2:** A common floorplan for switched capacitor circuits.

usually quite easy to apply for filtering applications. For such applications, typically, each op amp has associated with it a single capacitor bank, and a small collection of switches. These op amp-determined strip segment triplicates form natural clusters. Additional routing channels are generally placed between each such cluster for its own internal and other routing.[87] Such macromodule triplicates are usually treated as a whole in global placement strategies. These strategies may attempt to optimize the circuit by moving the triplicates around driven by traditional optimization criteria, e.g. minimization of total interconnect length. However, for switched capacitor filters, each of these triplicates typically corresponds to one of an ordered series of filter "stages," making their relative placement fairly obvious. Within an individual macromodule triplicate, ordering of switch cells within the switch strip, and arrangement of capacitors within the capacitor bank, is typically driven by routing considerations as will be described shortly. For the specialized but common forms of analog circuitry that can use this second, more specialized strip architecture, these being primarily switched capacitor circuits of various kinds, the difficulties in layout lie mainly in intra-stage routing and in low level (e.g. op amp) module synthesis.

The tools for global floorplanning and placement that employ the second, more specific strip architecture are usually themselves particularized for the unusually regular type of circuits that fit this layout structure. As was just noted, they are generally so simple, e.g. being guided by the ordering of stages in a filter's definition, that they gain little attention in the literature. On the other hand, with the more general first architecture, digital placement tools are often employed for global cell and polycell arrangement. Very often this is the case because the circuit as a whole consists of digital as well as analog cells, and the digital cells are standard cells. It is most convenient in this situation to simply employ a conventional digital standard cell placement tool to arrange all cells, both digital and analog. (The analog cells being preconfigured polycells where necessary.) Smith et al.[77] and Gruver et al.[51] describe a digital standard cell layout system adapted to accommodate analog cells in this way. Kim et al.[41] on the other hand, describes a cell placement algorithm specialized for analog layout. However, except for its use of the first, more general alternating-channels strip architecture adaptation, it is not materially different from a conventional digital placement tool. The recent review of Berkcan et al., (13), briefly surveys work in floorplanning and placement at the cell assembly level, including a few references to the European literature not covered here.

It is not surprising that simple techniques, either ad hoc or digital, are sufficient for cell placement at the cell assembly level of circuit integration, as this level is generally identified by its suitability for simple placement methods. As was noted in the introduction to this section however, analog cell assembly is not entirely as carefree as that of digital. In analog work, net isolation is still a major concern even for cell assembly. This generally complicates routing. As most crosstalk and parasitics, especially those that are hard to predict prior to layout, are the result of interconnections, net routing is very influential in determining and hence controlling these undesirable side effects of physical circuit realization. Hence routing between even large analog macro or polycells must generally still be carried out somewhat more carefully than it is for digital circuitry.

Routing is affected in several ways by the additional concerns of analog layout at the cell assembly level. To begin, for analog layout, minimization of fundamental capacitive and resistive interconnect parasitics entails reducing the lengths of interconnects. This is much the same as for digital routing. However, for analog layout, this reduction must generally be addressed on a per-net basis rather than for a circuit as a whole. That is, minimizing the sum total interconnect length of all nets as a whole is not a good strategy for optimization in analog layout. Parasitics on some nets are more critical than others, and can vary in importance depending upon their values. For example, a .01 pf increase in a parasitic capacitance may be negligible at some values of the total parasitic, but significant at others, even for the

same net. Also unlike digital routing, sensitivity of some interconnects to resistive parasitics may entirely preclude use of some high-resistivity interconnect layers from routing of some nets. The most important complication of analog interconnect routing, when matching is not required, however, is internet coupling. Ensuring adequate internodal isolation is generally accomplished by arranging for sufficient separation to exist between the nets involved. The separation distances required are generally larger than the minimum spacing rules employed for digital routing. In some cases, a few multiples of a minimum wire width may be entirely adequate, while in others it may be desirable to have the interconnects involved spaced as far apart as possible. Interconnect crossovers are especially notorious for inducing internodal coupling. They must be avoided entirely on some nets while being tolerable but undesirable on many others. All of these concerns make routing of analog macrocells a job distinctly different from that for digital cells.

The first step in automating an analog routing task is usually to impose the restriction that all routing be *channel* routing, that is, that no routing be permitted over already-occupied areas, even if allowed by the fabrication process. An interconnect placed over existing circuitry of any kind will be capacitively coupled to it. Thus over-module routing is generally avoided in analog layout. Although good analog layout professionals sometimes effectively take advantage of occasional opportunities in analog layout for over-cell routing, due to the difficulty in identifying these opportunities, they have in general not been exploited in automated analog routing. (However, see (22).)

Automatic channel routing is a highly developed technology in digital layout. Although some have attempted to employ automatic unaltered digital channel routing methods for analog layout,[7, 10, 25] for many analog designs, just keeping interconnect off existing layout is generally insufficient to control parasitics and crosstalk at acceptable levels. Therefore, the next step in automating analog routing typically is to impose a classification system on the circuit's nets that will facilitate a more refined treatment of the problem. These classification systems consist of a discrete, finite set of net "types" simple enough to be manipulated automatically, but expressive enough to capture the variety of special nodal uses, treatments, and privileges required for good analog layout. In general, they merely formalize informal net stereotypes commonly used in manual layout. In addition to "sensitive" nets requiring extreme isolation, and "noisy" nets which are particularly harmful, such a classification system may also include "neutral" nets of varies kinds, such as "power" nets (extensive and high current carrying), and "bias" nets (less extensive and/or having lower currents). Any or all of these primary net types are sometimes further appended with such designations as "critical" or "non-critical" (e.g. on sensitive nets), or "digital" or "analog" (e.g. on power nets, which are often segregated as such). A net's type must be assigned prior to layout. Usually this is done manually. (However, see section 3.3.) The purpose of net classification is to provide additional information to an automatic router or other layout tool so that it may better respect internodal isolation concerns where they are important, while at the same time exploiting available layout freedoms where they exist.

The additional information provided by net typing can be used in a number of ways to effect better analog layout. Its use in channel segregation, that is, assigning specific channels to specific types of nets, has already been discussed, as has its use in module clustering. Kimble et al.,[42] further utilize net typing to segregate tracks within their nonsensitive channels as shown in figure 3.

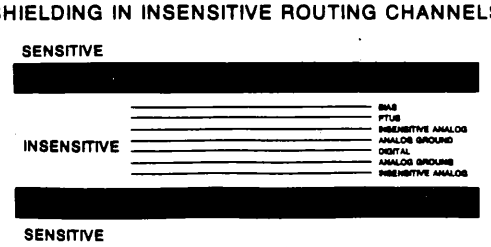


**Figure 3:** Net segregation within a channel by type (from (42)).

This scheme takes advantage of the ability of neutral bias and power lines to shield various signal net classes from each other. Another obvious application of net typing is prioritizing the routing process. In its simplest form, prioritization merely determines the order in which nets are considered for routing. Nets routed earlier generally enjoy less restricted routing conditions and can obtain more desirable tracks in the routing channel. Kimble et al.[42] for example, route nets from the edges of the channel inward toward its middle. The higher priority sensitive nets in this system thus experience fewer crossovers, as later through-channel routing is able to avoid them completely by utilizing only the as yet unoccupied portion of the channel near its center.

A higher priority can also mean that a net will be more influential in determining cell placement. Recall that, when employed, module clustering techniques aim to internalize sensitive and/or critical nets. Many systems that employ a clustering technique for net isolation also use routing concerns in defining more specific cell arrangement details within clusters as well. Under the classical switched capacitor floorplan of figure 2, for example, Yaghutiel et al.[87, 88] determine the order of switches within a switch cluster, and the order of capacitors within a capacitor bank, so as to minimize crossovers. Here, net type is used to give crossovers of sensitive nets priority for removal. Barlow et al.[8, 9] go a step further and classify some sensitive nets as critical. They then identify subclusters of capacitors and switches connected to these critical nets and place them together within their respective strips. Routing within each subcluster, in particular, routing of each subcluster's defining critical net, takes precedence over inter-subcluster routing and is performed first. This method, they claim, nearly always guarantees that the identified critical nets will experience no crossovers.

All of the techniques for analog routing described so far have been fairly ad hoc -- they do not make use of any widely recognized, general problem formulation or solution method. The next example represents a departure from this rule. It also anticipates a much larger body of work to be covered in section 3.2 on module generation. This technique makes use of a *constraint graph* to represent analog related routing restrictions, a structure, as will be seen, that has many uses outside of routing as well.

This example is a method of performing *detailed* channel routing for analog layout. Detailed channel routing concerns the selection of tracks for individual interconnections within a (usually elongated) rectangular channel having component connection points along it boundary. Detailed channel routing can be accomplished by applying a standard optimization paradigm in which a constraint graph representing all interconnect restrictions is first constructed and then *solved.* In this particular application, constraint graph solution means finding an assignment of terminal-to-terminal interconnect segments to channel tracks in such a way as to minimize channel width while at the same time respecting all restrictions represented by and contained within the constraint graph.

Chen and Kuh[20] describe a constraint representation and graph solution method for channel routing of digital circuits. In this scheme, graph nodes represent terminal-to-terminal interconnect segments, and an edge is placed between two nodes if the corresponding segments must share a common stretch of channel (and hence may influence each others placement). A directed edge between two nodes indicates that one of the corresponding segments must be either above or below the other (as indicated by edge direction). Undirected edges simply convey that the segments cannot coincide. The edges are weighted, with the weight indicating the minimum distance by which the segments must be separated. (See figure 4.) The kinds of routing restrictions that can be represented by constraint graph edges in this way are inherent to all channel routing problems whether analog or digital, as they are a natural result of the overlapping extents of interconnect segments that must be placed within the channel.

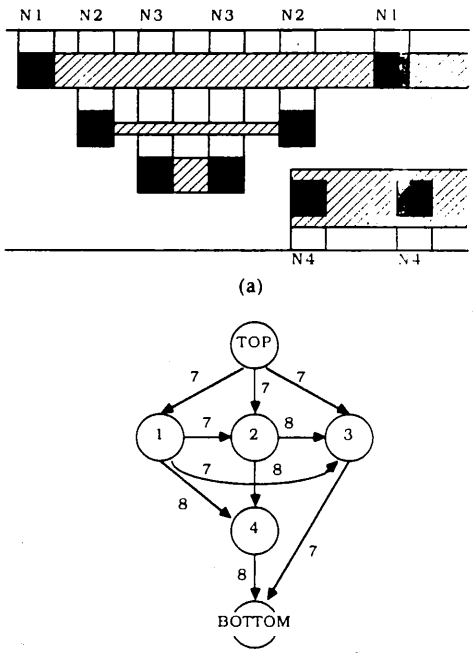


**Figure 4:** A detailed channel routing and its constraint graph (from (30)).

Gyurcsik et al.,[30, 38] recognized that net crossovers, layer restrictions and proximity limitations, all important concerns for analog layout, can all be enforced by imposing these simple ordering and spacing constraints on the wires connecting terminals along the edges of a routing channel. What they have done is to add additional edges to the constraint graph, before it is solved, to represent these analog layout restrictions. Crossovers are excluded by requiring one of the involved segments to lie above or below the other. To avoid high resistivity

routing layers and/or inter-layer contacts along a given interconnect segment, the segment can be constrained to lie on just one layer. This can be accomplished by forcing all other proximal interconnects to either lie above or below the constrained one. Proximity limitations are enforced through proper definition of the weights. Thus by the simple addition of more specific constraints, the same technique described by Chen and Kuh can be used to generate a channel routing plan that minimizes channel width without violating any indicated net crossover or proximity restrictions.

Because this technique does not attempt to reorder the terminals along the channel's boundary, and can space any two nets according to any isolation criteria, it can be applied without the use of a channel segregation or module reordering mechanism. However, it cannot entirely replace them, as for some module placements the constraint graph may not be solvable. If this were the case, some module reordering would be required. Additionally, much waisted area could result if nets requiring substantial separation distances were required to share the same channel rather than being allowed to occupy two widely separated ones. Nevertheless, this technique represents a definite advance in the application of systematic methods to the often perplexing plethora of concerns in analog routing. Several more examples of methods based on the use of a constraint graph to represent analog layout concerns will be presented in section 3.2.

This last, more elaborate constraint-based analog routing method has apparently not yet been used in any actual design tools or design applications. However, two of the techniques mentioned before it are in fact either in commercial use or competitive with systems that are in such use. The system described by Kimble et al. has been in production use since 1983,[42] while Barlow et al. report that their system consistently produces area densities within fifteen percent of intensive hand-packed layout.[9] From these reports, one can conclude that sequentially prioritized net routing and routing directed placement schemes, the techniques these systems use, when properly applied, are apparently adequate, at the module assembly level of integration, for many analog IC layout tasks. It can also be observed that all of the routing methods described in this section are based on some digital technique which has been augmented to take into consideration analog routing concerns through use of a simple net classification scheme.

### 3.2. Module and device generation

Although assembly of large fairly complete analog building blocks can be difficult, as was seen in section 3.1, the real challenge in automating analog IC layout is in integration at and just above the device level of component complexity, that is, in properly arranging and interconnecting no more than ten or twenty electrical devices in a compact, performance maximizing way.

There are several reasons why layout at this level of complexity can be difficult. First, it is, of course, still necessary to worry about parasitics and isolation. In fact, a larger proportion of the nets involved usually must be watched for proper isolation. Moreover, because there are fewer components per net, there are more opportunities to implement connections through direct abutment or short, two-terminal routing segments. This makes the difference between good and bad component placements more extreme when it comes to minimizing net parasitics and crosstalk, and thereby puts a premium on proper placement in controlling these adversities. Secondly, placement is complicated by the fact that the components to be integrated, individual devices or small subcircuits, do not, in general, have geometries amenable to any simple layout plan, such as a strip architecture. For individual devices, it is much more the case that form is determined by function. For instance, for small devices, terminal locations and aspect ratios, the ratio of a cell's length to width, can be restricted by performance limitations. A more significant restriction however, is that the area of a cell for a device is usually determined by its electrical specifications. This results in a third complication, as the areas of cells often vary substantially, sometimes by as much as two orders of magnitude or more. This size variation can make efficient utilization of area difficult. Lastly, there is, of course, often the need for symmetrical layout.

The needs for symmetrical layout and the large variations in device sizes virtually preclude adaptation of established general purpose digital synthesis methods to this type of analog layout. Digital layout systems never attempt to match wire dimensions when routing, as would be required for symmetrical layout, and for placement purposes, these methods often idealize all components as being of equal size. Thus analog layout at small levels of integration is fundamentally different from both analog layout at larger levels of module complexity, and digital layout at all levels. In general, it is more difficult than any of these other layout tasks.

Due to the extreme difficulty of low level analog layout, most early attempts at automatic analog layout synthesis concentrated on module assembly, leaving the more exacting internal layout of the cells that were to be assembled to manual methods.[42, 51, 77] However, this still left a substantial amount of layout to human effort, as every new analog IC design still required many new analog modules to be laid out. Repeated attempts to develop libraries of widely applicable, general purpose standard cells for analog design, akin to those so successfully employed in digital IC layout,[40, 47, 50, 82] have generally failed.[15, 17] The libraries never seemed to contain a cell close enough to what was required to meet specification. Instead, each new analog circuit design generally entailed adding new "standard" cells to the library, cells that would inevitably end up only being used within that original application.[17] It is not hard to see why this might be the case. The most important details distinguishing any two analog circuits are at the device and small subcircuit level. Indeed, the use of general purpose circuit *topologies* that can be easily adapted to different uses through device sizing alone is, in fact, a common practice in filter design.[27, 54] Thus it is not uncommon for analog circuitry of considerable functional diversity to differ in construction *only* in the specifications for the devices used. Hence, the only fixed cells that have found truly wide application in analog layout have been those whose internal device sizes do not significantly determine circuit function, such as digitally controlled analog signal switches.

### 3.2.1. Procedural methods

What has proved much more successful at accelerating low level analog layout than attempting to maintain libraries of reusable but individually designed standard cells, has been the use of parameterized *cell generators.* These are computer programs or subroutines that accept a handful of circuit specifications as input and produce as output a layout for a specific device or circuit that meets the given specifications. Because it relies on conventional procedural programming practices, use of parameterized cell generators is also referred to as *procedural specification.* Usually, the input parameters of a cell generator correspond very closely to geometric characteristics of the desired layout, such as device sizes. If the generator is to produce a multi-device circuit, usually the circuit's topology and floorplan are largely fixed.

Although developing a computer procedure for generating a certain class of layouts is generally more difficult than directly implementing any one of its possible outputs by hand, unlike analog "standard" cells, analog "standard generators" have proven genuinely reusable. In fact, due to the importance of precise device sizing in analog IC design, as well as to the relative simplicity of device geometric forms, cell generators for individual circuit devices have been employed in analog layout for years and are now quite common.

Allen et al.[2-5, 75] pioneered the use of direct procedural layout specification for multi-device analog layout, but writing computer procedures to automatically generate complex circuit layout did not become popular in analog design until this layout description method had become more widely used in IC design automation in general, particularly, not until *digital* IC layout had become so complex that procedural methods were needed for its automation as well. Once this occurred, many programming tools specialized for IC layout became available that could be used or easily adapted for analog layout. In particular, compilers for special purpose programming languages were developed specifically for procedural layout specification. These languages, for instance, made recurring operations in layout specification, such as component placement by relative positioning (e.g. *above, right-of* ) and interconnect path definition easier and simpler to describe. Kuhn[46] describes the use of a special purpose layout language for analog layout. (See figure 5.)

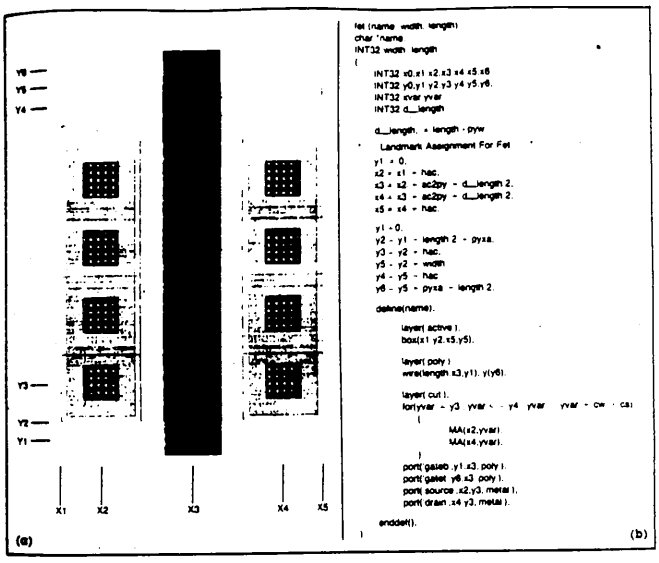


**Figure 5:** A sample layout and its procedural specification (from (46)).

Since parameterized cell definition is based on contemporary computer programming methodology, it is in principle completely general and capable of being applied to any layout task large or small. Automatic generation of layout from procedural specification is often referred to as *silicon compilation.* Use of such methods is in fact growing in both digital and analog layout automation. Both Kuhn,[46] and Helms and Russel[32] describe very highly developed silicon compiler systems for analog layout. The Concorde system[32, 33] for example, contains a library of generators for complete A/D and D/A converters, for switched capacitor filters, and for all the principle components of these circuits, such as op amps. It is claimed that these systems produce modules with performance comparable to hand crafted design.[34]

Procedural layout specification has not proved to be a panacea for analog layout however, as it merely trades one type of complexity for another. Programs for generating special purpose cells are many times more difficult to develop than any one of the circuits they produce would be. Moreover, the disparity between the two levels of development difficulty grows exponentially with the complexity of the circuit under consideration. Hence, in general, it is only economically justifiable to develop procedural specifications for very simple layouts, such as those for individual devices, for highly regular layouts, such as those often found in switched capacitor filter designs,[7, 69, 70] and for other complex circuitry only when it is expected that the generators developed will be very highly used.

### 3.2.2. Template based methods

One of the reasons procedural layout specification is so difficult is that it requires the layout to be described in every detail all at once. It is difficult with procedural specification, as with any other programming task, to operationally describe a high level or abstract general plan without including along with it many of the incidental low level particulars that would make the plan into a complete design. It is also not possible with conventional programming methods to describe the details without specifying an often irrelevant order for their addition to the design. Thus for IC layout, detailed, sequential, step-by-step construction commands can be an awkward, inefficient way to declare what is in fact a static object. This is especially true when the design conforms to a large number of uniformly applicable constraints, that would thus have to be repetitively checked or enforced throughout the specification. Yet integrated circuits, do conform to a large number of such constraints, as the many *spatially* uniform electrical and manufacturing restrictions are constraints of this type. Thus procedural layout specification is something like trying to describe a picture in words. It can often take a lot of words to describe even a simple picture.

The static and regular nature of IC layout has long been exploited for simplifying digital layout specification. It is this set of characteristics that make practical various "stick model," "virtual grid," and "symbolic" layout specification techniques widely used in digital design.[71] Each of these layout specification methods utilizes a static, non-procedural geometric description format much like an actual layout. However, it is not necessary, when utilizing one of them, to decide exact inter-component spacings or intra-component dimensions. Rather, these fabrication technology determined particulars are included later by an automatic tool usually called a *spacer* or *compacter.* The spacer combines the imprecise layout description with definite spacing requirements for a specific fabrication process to generate a completely detailed layout description. Thus, these methods separate details particular to a given application from details specific to a given fabrication process. In doing so, they achieve both a reduction in complexity of either set of details considered separately, while at the same time increasing the overall generality and hence usefulness of each set of details. For instance, the same "stick model" layout description can be combined with many different technology spacing requirements to generate many different layouts.

Much the same idea can be applied to analog layout. However, the factorization of design details into less complex, more general subsets is, in general, more usefully accomplished somewhat differently. The features of a layout particular to a given analog design are more extensive and varied, hence the symbolic abstraction chosen to represent them must be more elaborate and flexible. Also, providing independence from the fabrication technology is usually not the primary motivation for adopting such a specification strategy. Rather, the impetus is usually to make design features common to many applications readily accessible and easy to combine with those that are not. (As somewhat of an exception to this general rule, (63) and (55) describe attempts at developing general purpose analog compacters modeled after those employed for digital work.)

In analog layout, the abstract, static layout specification is usually referred to as a *template.* Like their digital counterparts, the purpose of such a data structure is to collect together features of a layout

that must exist together for some purpose, and to perform this collection function without overly restricting the variety of applications to which the data so collected might be applied. For analog work, this usually means ensuring proper net isolation and matching, while at the same time allowing large latitudes in device sizes. Analog templates are usually specific to a given circuit topology. A single circuit topology, on the other hand, may have a large library of templates associated with it, each being specialized for a particular range of device sizes or other circuit characteristics. In general, a template can contain almost any type of layout information, from complete geometries for certain "seed" portions of a larger form,[69] to partial routing plans.[83] However, they are most commonly employed to map a sized schematic (i.e. a circuit topology with device sizes) into a suitable floorplan.

For definiteness, we will describe two floorplan description formats that have been used to describe layout information in a template format for analog IC design. Both of these formats have been borrowed from digital design automation, but are not as popular for digital work as they are in analog applications.

The first of these formats is the slicing structure.[67] A slicing structure describes the positions of components and collections of components relative to "slicing" boundaries. A component or a collection of components is represented in a slicing tree by its bounding box. In describing a floorplan, each such bounding box is declared to lie entirely to one side or the other of a slicing boundary, that is, either *left of* or *right of* if the boundary is vertical, or *above* or *below* if the boundary is horizontal. The collections of components on either side of a slicing boundary constitute a new, larger collection, with its own bounding box, that may again be joined to others along any of its bounding edges (see figure 6).

Although the result will not always be area efficient, the bounding box adjacency declarations in a slicing structure specification will always produce a floorplan for the components involved, regardless of their dimensions. The virtual grid layout specification format, which is more popular in digital layout, also has this same ability, however, its area efficiency is usually much poorer for collections of components that vary substantially is area. Besides varying within a single application, sizes of individual analog devices for the same topology can vary substantially from application to application, making exact rather than relative positioning difficult to anticipate. This accounts for the greater popularity of the slicing

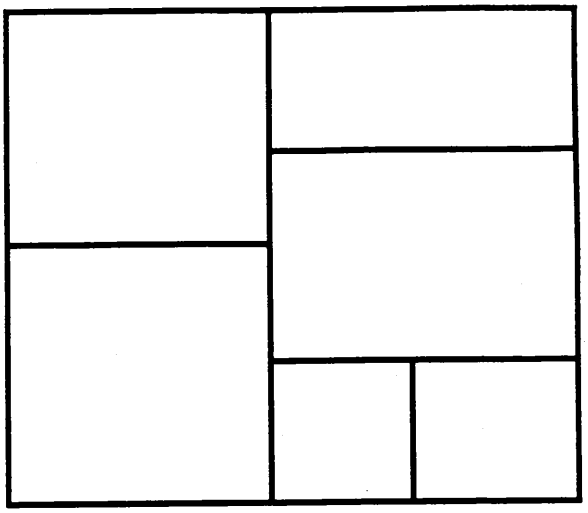

**Figure 6:** A slicing structure floorplan.

structure in analog layout specification. The hierarchical structure of a slicing structure makes this description format compact, but a slight price is paid for this economy, as not all possible arrangements of groups of components are describable by a slicing structure (see figure 7).

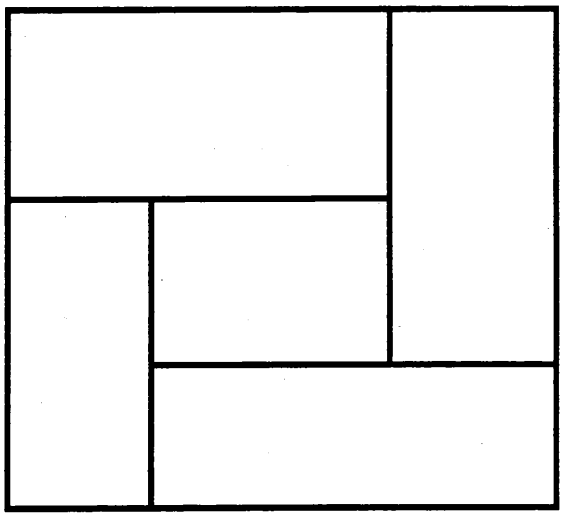

**Figure 7:** A component placement not representable by a slicing structure.

Koh et al.[44] describe a simple application of the slicing structure to template-based floorplan specification for analog layout. In their system, an automatic topology selection and device sizing system searches a database of slicing structure floorplan

templates for an appropriate fit for the given topology and device sizes. This floorplan template determines the relative positions of each component. Once the floorplan has been selected, layouts for the individual devices are generated procedurally and placed within it, and the circuit is automatically routed. Then the entire layout, still represent in a loose, symbolic format, is processed by a standard digital compacter to determine precise inter-component and wire spacings. In this system the template also includes information forwarded to the compacter so that it will respect various circuit-specific spacing requirements, such as unusually large separation distances.

The second floorplan description format often found in templates for analog layout is not limited in the way that a slicing structure is. It can, in fact, describe all possible floorplans. It involves direct specification of all component adjacencies in the form of a *constraint graph.* When used for floorplan specification, the nodes in a constraint graph represent edges of component bounding boxes. They may also represent edges of bounding boxes of collections of components or any other meaningful geometric datum, such as a slicing boundary. Two nodes are connected by a graph edge if the layout edges corresponding to these nodes restrict each other's placement. Usually the edges are directed and weighted. Edge direction indicates which layout edge must lie to the right of or above the other. The edge weight usually represents a minimum, or sometimes a maximum separation distance.

In digital layout, constraint graphs are often used to direct the operation of an automatic compacter, which attempts to minimize the area of a layout by positioning its components as close together as possible. The constraint graph provides the geometric layout information it needs to do this. In these digital applications, the constraint graph is usually constructed automatically from an initial, uncompacted circuit layout, which supplies the graph nodes, edges, and edge directions. A separate description of manufacturing tolerances, called "design rules," is used to derive the edge weights. This is in fact the method employed by the compacter used in the system of Koh et al. In general however, for analog layout, it is not as easy to determine edges from a single initial placement, since component size variations can make such a rough placement just as difficult to come by as a good one, and, in any case, the spacing requirements represented by the weights are not circuit independent, but can be a function of circuit topology and many other design specific particulars. Indeed, it is for these reasons that floorplans are usually incorporated into templates for analog layout.

Mogaki et al.[55] for example, describe a compacter intented for general use over a range of analog circuits. Using pattern matching techniques, it selects a template containing the necessary constraints from a library. The pattern matcher compares features of the inputed circuit topology to keys stored with each library template. Like the system of Koh et al., the selected templates contain little layout related information save conditions describing relative component adjacencies, and the actual layout definition is produced by an essentially conventional, digital compacter.

A constraint graph is actually a very general purpose description format whose use in layout specification is not limited to adjacency declaration or floorplanning however. Because constraint graphs describe relationships between component edges, they can also be used to describe the internal structure of components as well as their position with respect to one another. (See figure 8.) If nodes are allowed to represent any quantity, not just the locations of layout edges, and these quantities, along with the graph edge weights, are allowed to be defined by mathematical expressions, the result is a more complete methodology for specifying layout. In expressive power, this specification methodology is comparable to procedural methods, however, it itself is not procedural but static, or *declarative,* as there is no notion of sequential ordering or of time. When used for general software development, description in terms of static, declarative constraints is referred to as *constraint programming.*[49] In this paper, this type of layout specification will be called *constraint-based.* The descriptive freedom possible with constraint-based specification has generally required that a formal language be defined for its expression. To date, all such languages developed for integrated circuit design have been textual, patterned after procedural programming languages,[12] although it has been recognized[49] that a graphical entry format would be more desirable for geometric applications.

Constraint-based specification is used in digital as well as analog layout,[1] but seems to be more popular for analog work where the simpler, less flexible methods more common in digital design cannot adequately express or easily describe the many and varied interactions that may exist between details of an analog layout.[8, 9] Berkcan et al.[12, 13] for example, have reported what is possibly the most elaborate constraint-based system to date. It is intended primarily for analog IC design. The constraints in this

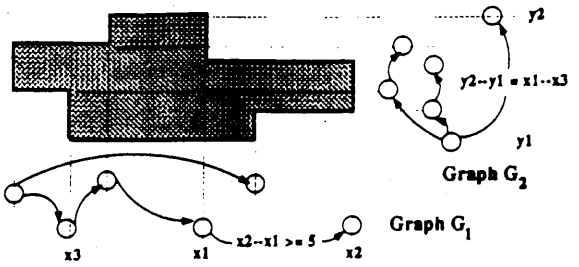


**Figure 8:** Constraint graphs describing a polygon (from (14)).

system may relate quantities defined by very elaborate mathematical expressions. A general purpose numerical optimization method is employed to derive the layout geometry from its specification. This system also allows hierarchical specification, permitting, for example, a template for a large circuit to be defined in terms of separately declared templates for its subcircuits, much as a subroutine in a procedural programming language can be described using other subroutines.

Like procedural specification, template and constraint-based layout description methods attempt to transfer the complexity of full and general layout description into another, more tractable form. The static, often geometric format of template and constraint-based specification is more efficient for layout than is procedural declaration, but such specification methods are still very labor intensive. Although more general than specific layout examples, the ranges of applicability of many template or constraint-based descriptions can still be fairly narrow.

### 3.2.3. General approaches

All of the strategies for automatic analog IC module generation described up until now can be considered generalizations of the standard cell design approach. Although they do not rely exclusively on fixed geometries for layout definition, they nevertheless accede to the belief that at least some geometric layout details can only be provided on a module-specific basis if competitive results are to be obtained. A truly circuit independent automatic layout system would naturally be much more efficient, obviating the need for customized procedural encoding or template definition an a module by module basis. Such an automatic synthesis system would also be more complex and difficult to develop, so much so that it has not as yet proved feasible to design such a system as a single, integrated whole.

All general purpose analog layout synthesis systems that exist to date consist of a collection of independently operating subsystems. Not surprisingly, despite the closer interaction between placement and routing in analog layout, efforts to develop circuit-independent automatic layout systems for analog ICs have for the most part followed traditional, digital practice in dividing the problem into the two independently conducted tasks of placement and routing. Hence work on these two problems in analog layout can be described separately.

For circuit-independent placement, three strategies have been attempted to date. The first, which might be called the heuristic approach, attempts to derive an IC floorplan through application of some direct, systematic and usually ad hoc algorithm. A contrasting second approach has been to formulate the problem as a kind of search or optimization, and apply a more general algorithm to solve the problem so formulated. As the third approach, somewhat midway between the previous two, are various *knowledge based* methods.

Knowledge based methods can be viewed as either general or ad hoc techniques. They are general in their utilization of a standard expert system, rules-plus-inference-system software architecture, but ad hoc in their application of special purpose rules or "knowledge." These systems take as their cue the observation that most analog IC layout work accomplished today is indeed done by specialized experts, thus making the job a likely target for expert system automation methods. Most knowledge based approaches to analog layout are able to subsume the task of routing along with floorplanning in their framework. To date however, most efforts in this direction have resulted in the development of only a general automation design framework that does not yet include sufficient expert knowledge to generate anything substantive. Most of these efforts have thus, for the most part, produced more promise than product. One of the more substantive examples, that of SALIM,[39] however, will be described in some detail in section 3.3.

Heuristic type placement methods attempted to date are based on two ideas, both of which were first mentioned in section 3.1.

The first idea is to derive the information needed to identify a suitable arrangement of components by inspecting the organization of the circuit's schematic. It is very often true, especially in analog IC design, that, when represented graphically, the geometric arrangement of circuit symbols in a manually entered description of a circuit topol-

ogy often strongly resembles a good overall floorplan. This can be particular true with regard to relative positions of matching devices and to sensitive net and power line routing. However, even were this not already so, it would still be very simple to *adopt* it as a design procedure, that is, intentionally arrange the symbols and interconnections in a circuit schematic to reflect a particular layout floorplan. Such an effort could in fact also easily be extended through the use of schematic *annotations* which indicate additional information not required for topology definition, but which are useful for guiding an automatic layout tool.

A tool employing annotated schematics in this way has been reported by Mehranfar.[52] In this system, the relative positions of the device symbols in the schematic are used as an heuristic metric with which to guide the placement of device geometry in the actual layout. That is, the relative closeness of symbols in the schematic is taken as a preference as to how close the corresponding cell geometries should be placed in the layout. Placement is bottom up, with smaller components being merged into larger components before the larger ones are examined, and components having matching constraints being placed first. The components that must match are identified from schematic annotations (see figure 9).

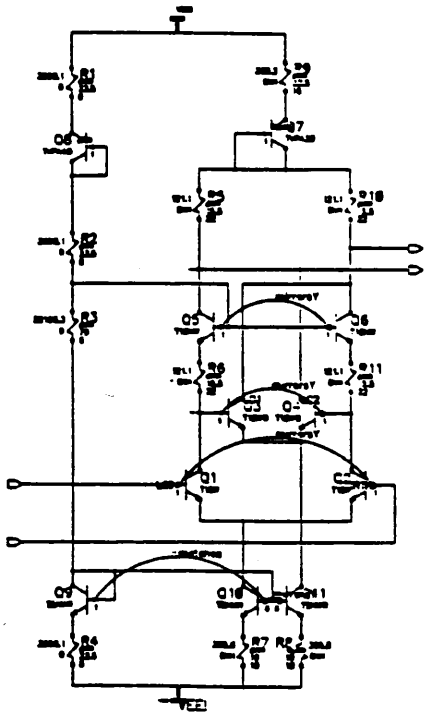

**Figure 9:** An annotated schematic (from (52)).

Ito and Mori[37] utilize a more direct procedure. They map the schematic directly into the plane of the layout in such a way that the actual dimensions of each device's geometry are small in comparison to the size of the symbol that represents them in the schematic. This ensures that their will be no overlap after the second step, which is to replace each symbol in the schematic with the actual geometry of the device, and to locate interconnect wiring at the precise locations of the interconnection lines shown in the schematic (see figure 10). This leaves them with a complete layout for the circuit that can then be reduced to a more reasonable size by a standard compaction step.

The schematic inspection approach is not quite module independent, as the schematic itself is merely serving as a template for the layout desired. However, as a circuit schematic is usually constructed for documentation purposes independent of layout, the method at least requires little or no *additional* information to be provided by a human user.

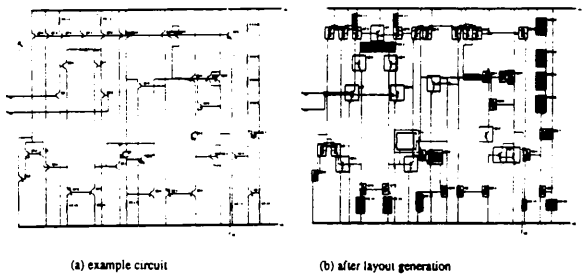


**Figure 10:** Substitution of device geometries for symbols in a schematic (from (37)).

The other heuristic approach for general, module-independent placement is based on analysis of interconnections. With this strategy, the relative proximities of components is derived from the number of nets of various classifications (e.g. sensitive, insensitive) that the components share. It is the same idea used to cluster modules for assembly described in section 3.1, however, now the actual geometric arrangement of the components is determined, not just groupings. The schematic directed system of Mehranfar[52] just described can be configured to alternately utilize this approach instead. To do so, they simply replace the schematic derived proximity metric with one determined by the locations of already placed components. The metric measures the interconnect lengths from these previously placed objects to the current one, with different interconnects carrying different weights based on

their net type. Winner et al.,[85] as mentioned in section 3.1, identified clusters of related components by tracing out sensitive net trees until they terminate at components having no sensitive nets not already in the tree. After cluster identification, they then arrange these matching-height components in a strip so as to minimize wire length among the identified sensitive nets. This second placement approach based on net analysis is much more module independent than the schematic inspection approach. The input circuit topology need include no geometric information and could thus in principle be generated automatically by a separate topology synthesis tools.

Regardless of the method, whether or not an ad hoc heuristic type placement strategy, such as those just described, actually produces results that are as good or nearly as good as they could be must usually be taken on faith. The type of placement approach that formulates the placement task as another more general search or optimization problem however, often can rely on general solution algorithms for which the ability to produce good results is better supported.

All the general search or optimization based placement methods that have been tried to date for analog layout formulate the problem so as to be solvable through *simulated annealing.* Simulated annealing[43, 74] is an iterative improvement optimization technique that searches a space of solutions by repetitively modifying and updating a given one. The relative desirability of each solution is measured by a numerical *cost function,* the objective being to find a solution with as small a cost as possible.

The ANAGRAM system[26] represents a straight forward application of simulated annealing to analog placement. In this analog IC layout system, geometries for devices and small subcircuits are generated procedurally. The job of the simulated annealing floorplanner is to determine a suitable arrangement, orientation and aspect ratio for each of these low-level procedurally generated cells. The modifications, or *moves* employed by this simulated annealing implementation consist of changing a cell's orientation, aspect ratio, or location. The cost function is a weighted combination of terms measuring total layout area and an estimate of total wire length. Because moves in this system can cause cells to overlap, an impossibility for any workable floorplan, the cost function also includes a "penalty term" measuring cell overlap and thereby flagging configurations having overlap as undesirable. A more recent version of this system, ANAGRAM II,[22] enforces symmetry requirements by forcing all moves involving either one of a pair of matching components to be appropriately reflected in the other.

By employing a slicing structure representation, Rijmenants et al.[72, 73] obtain a much more efficient application of simulated annealing to floorplanning and component placement, in a system called ILAC. Since cell positions in a slicing structure are always relative, use of a slicing structure avoids completely any impossible states caused by cell overlap. Moreover, it alleviates the need to directly represent cell aspect ratio in the solution space. Instead, aspect ratios can be made a function of relative position, as the optimum aspect ratios of all cells can be systematically determined from the aspect ratio imposed on the layout as a whole, the position of the cells in the tree, and from the *shape functions* of the cells. A shape function describes a cell's achievable width and length combinations. As in ANAGRAM II, symmetry requirements are enforced by coordinating moves on matching components. The greater efficiency of the slicing structure approach makes it practical to spend more computational resources evaluating the desirability of tentative solutions. In fact, in ILAC, floorplans can be completely routed before they are evaluated. Thus the wire length and area "estimators" used for solution evaluation are very accurate, and the cost function can include penalty terms for net crossovers and other routing related undesirables.

Simulated annealing is a very computationally intensive method, typically involving consideration of many thousands of possible solutions. For analog floorplanning, execution times are generally measured in minutes, not seconds, even for fairly simple circuits. Current heuristic methods however, lack the ability of simulated annealing to simultaneously consider many diverse factors, from layout area to net crossovers, in choosing a solution. Moreover, unlike many other general purpose optimization techniques, the cost function for simulated annealing is completely unrestricted. It is used in the algorithm only for evaluation, and need not be linear, convex or have any other special properties in order for the algorithm to work, at least in principle. The ability of simulated annealing to accommodate many types of concerns simply through inclusion of appropriate terms in its cost function is probably the principle motivation for its use in automatic analog layout, where the list of concerns is indeed long and varied.

In whatever way it is generated, once a placement has been chosen, the remaining task necessary to complete the layout is routing. As mentioned in section 3.1, to avoid unwanted coupling, over-component routing is generally disallowed, so that all routing must occupy channels surrounding the com-

ponents. Routing within a single channel has already been covered in section 3.1. Routing within an interconnected network of channels, one type of *global* routing, is similar in many ways. The details of net ordering within each channel are, in fact, essentially the same. The primary challenge new in global routing arises from the possibility of having multiple paths through the channel network between any two points. A sequence of channel segments through which to run each interconnect segment must then be chosen from among the multiple possibilities when such choices exist.

Although the issues that distinguish good global routings from bad ones are different for analog circuits than they are for digital ones, nevertheless, most of the tools reported to date for global channel routing of analog ICs are based on methods originally applied to digital layout. For the most part, the methods so adapted from digital layout have generally been based on an iterative search procedure, as these can often be modified for analog layout simply by including additional terms for analog concerns in a cost function. A most straight forward example of this adaptation method is provided by Koh et al.,[44] who employ a unmodified digital rooting tool for which the cost function can be user specified.

A somewhat more involved adaptation is presented by Cohn et al.[22] Their routing strategy employs a type of best first search. Nets are routed sequentially according to a net-type determined priority. For each net, a starting point is selected and a set of partial paths emanating from it maintained in a prioritized heap. The algorithm proceeds by selecting the most promising partial path from the heap, expanding it into several alternatives, returning these to the heap, and repeating the whole, selection, expansion, prioritization process until the lowest cost path selected from the heap is found to be complete, not partial. In digital implementations of this method, the cost function used to organize the heap might only consist of a sum of the path's current length and an estimate of the length of its completion. The adaptation for analog routing adds to this sum penalties for net crossovers, adjacencies, and other unwanted features. This tool can route a typical opamp is a few minutes. Note that this method is somewhat computationally intensive, and might not be practical on large circuits containing perhaps a hundred or so nets or components. However, this would not be a significant weakness for analog layout applications, where component and net counts rarely climb so high within independently designed cells.

Another search based routing technique is described by Piquet et al.[68] They sequentially route individual terminal-to-terminal connections. For each of these connections, a large set of possible complete paths is generated and then paths with properties undesirable for analog routing are pruned away. The novelty of the approach is that the concerns used to prune the set are also prioritized. The more important considerations are applied first, assuring their domination over others. Thus, if the set of possible paths should reduce to just one element before all undesirable effects have been accounted for, those concerns that may not have been considered will be the least important. This effect is hard to achieve by adding penalty terms to a single cost function, and it corresponds more closely to how multiple competing concerns are normally treated in analog layout.

For digital layout, routing is such an extensively developed technology that these and other reports[52, 73] hardly exhaust the possibilities in analog layout just for adaptations from the digital technology. For a recent survey of digital routing methods see (71).

### 3.3. Layout optimization

As was noded in section 2, analog IC design is most often a very empirical, exploratory endeavor, involving much trial, evaluation and redesign. Thus coordination of separately performed functions in analog IC design, that is, *system coordination,* usually takes the form an an iterative search, or *optimization,* involving solution generation, evaluation and modification. The association of system coordination with design optimization is so prevalent in analog IC design that no attempt to distinguish between them will be made in what follows. For the most part however, concern will center around optimization. Efforts to automate optimization and coordination processes in analog IC design that involve layout vary as to optimization criteria, extent of design coverage, and technique employed. It will prove most illuminating to begin with the simplest, least inclusive achievements, and progress to the more complex and inclusive but more challenging.

Before going on to describe even the simpler iterative optimization oriented coordination schemes, it should be noted that in some restricted domains, when performance requirements are not difficult to meet, a direct, non-iterative approach can sometimes be employed. That is, an essentially deterministic algorithm can be applied to a set of input specifications to yield a complete design, including layout. Although the result generally will not be as

good as could be obtained through a more thorough, exploratory effort, it might nevertheless be adequate for some tasks, and thus the approach justifiable on the basis of design economy. Naturally, such systems employ procedural encoding, and operate extremely fast. Total design times are typically measured in seconds, minutes or, if human interaction and some overall optimization are included, at most days.[2] This is a popular approach for switched capacitor filter design, because the simplicity of specification and regular structure of these circuits makes their synthesis relatively straightforward. Many such switched capacitor filter silicon compilers have been reported.[4, 25, 32, 87, 88]

To now begin development of iterative, optimization oriented coordination methods, the simplest optimizations that can be performed involving layout concern only this phase of the design process. They accept geometrically interpretable design optimization criteria, and effect only physical aspects of the design. Several optimization techniques of this type have already been described in previous sections. Two examples of the use of simulated annealing were presented in section 3.2. ANAGRAM[22, 26] used simulated annealing to optimally choose a floorplan to minimize total layout area and total wire length. This floorplan, once generated however, was not included in any other, more inclusive automated optimization scheme. ILAC[73] on the other hand, optimized floorplanning together with routing by including routing within the simulated annealing process. With routing contained within the optimization process's design coverage, routing related concerns, such as number of crossovers, were included among the optimization criteria.

Constraint-based specification, described in section 3.2.2, was yet another example of an automated design strategy based on optimization. In most cases, a constraint-based design description is *under constrained,* meaning that many design solutions exist that will not violate the given constraints. Thus, along with them there must generally be supplied an optimization criteria, such as minimization of total area, to select among the many possibilities. This in effect casts the whole design specification into the form of a classical constrained optimization problem.

Most small scale optimization problems, such as those just described, are usually formulated over either a discrete or a continuous state space. Simulated annealing is best suited for optimization over solution spaces containing discretely separated possibilities, while most constrained optimization solution algorithms concern the selection of continuously variable parameters. Since much of analog IC layout can be formulated in either a continuous or a discrete way, both of these approaches have ample applications in this field. For a survey of the use of simulated annealing in (essentially digital) VLSI design see Wong et al.[86] For a survey of numerical methods for parameter optimization in IC design in general see Brayton et al.[16]

Kayal et al[39] describe an approach to IC layout that can, in principle, accommodate either discrete or continuous state space optimizations. To do this, they employ a rule-based expert system architecture. In this system, called SALIM, layout tasks, such as floorpanning or routing for which efficient procedural algorithms are available, are implemented traditionally as independent tools in a conventional programming language. Operation of these standard, procedural tools, however, is controlled by a rule-based supervisory system that schedules their execution and evaluates their results. (See figure 11.) By pattern matching rule antecedents against characteristics of subtask outputs, important features, both good and bad, are identified for which corrective or otherwise special actions should be taken. For example, an undesirable crossover identified in the output of a router may result in re-execution of the router or floorplanner with additional constraints included in its input. SALIM's design allows a rule to perform any action whatsoever with regards to layout modification, so that, for example, a complete routing function could in principle be implemented within the rule-based part of the architecture itself solely through the use of simple, pattern-matched, if-then case-action rules. This high level of generality is offset, of course, by the need to manually formulate all the necessary rules. This system presents a good example of how system coordination and design optimization can be inextricably combined in analog layout.

Except for total layout area, which can be directly related to total cost, geometric optimization criteria are seldom a part of an analog IC's fundamental design goals. Nevertheless, since many of the original design goals that are specified can be highly influenced by physical design, it is generally useful, if not essential, to translate these higher level design objectives into geometric terms so that they may be properly respected, and may guide optimal layout generation. As it concerns the interaction between layout synthesis and topology selection, this translation process is an aspect of system coordination. If it could be automated, this would allow automatic layout generation procedures, particularly optimization

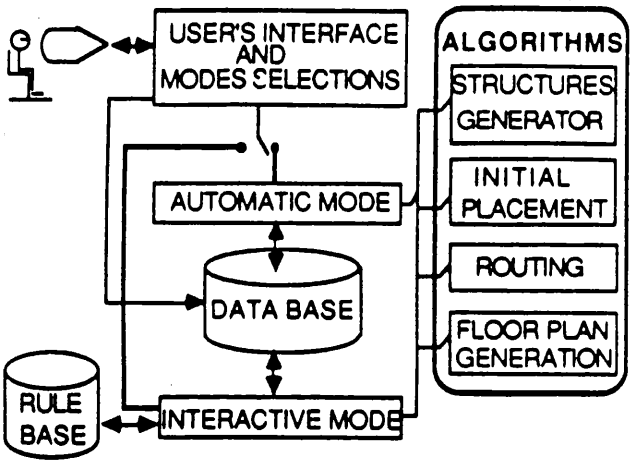


**Figure 11:** High level control structure of SALIM (from (39)).

routines, to operate directly with the original design criteria.

We have already seen how higher level design objectives are typically translated into more geometric forms. Once a circuit topology has been selected, its nets can be classified according to their sensitivity, components tagged with matching requirements, and so on. Net classification and matching requirements may then be interpreted geometrically by layout procedures as crossover restrictions, relative location requirements, and so on. For automation purposes, requirements such as net types and matching restrictions can be manually entered and stored in templates associated with each topology, thereby permitting automatic retrieval along with topology selection, or better still, generated automatically, directly from the topology and original design goals without manual intervention.

Hong and Allen,[35] and Gyurcsik et al.[31] report a technique for accomplishing the latter, namely direct translation from a topology and original design goals. In each of these proposals, a sensitivity analysis is performed on a selected, sized topology using a SPICE-level circuit simulator. In this way the relative variability of key performance measures to such circuit properties as component mismatch and inter-net coupling is assessed. These findings are then used, by Hong and Allen to automatically assign net sensitivity classification, and by Gyurcsik et al. to generate more specific net-to-net and component-to-component coupling and matching constraints. In a similar work, Choudhury et al.[21] describe a method by which SPICE determined sensitivities can be translated into constraints for a router.

With such techniques, a layout optimization procedure can indirectly utilize more fundamental optimization criteria, but the extent of influence of the optimization process would still be restricted only to the layout itself. That is, the optimization process could not trade off geometric considerations such as crossover reduction for more fundamental design features, such as device sizes. Design features like device sizes could not be influenced in any way at all by layout effects.

To see how design aspects such as circuit topology and device sizes can be optimized together with their geometry, it will be necessary to take a step back and review how these design features are determined in the first place.

As was noted in section 2, given a set of performance specifications, or design goals, in general, the first step towards completing an analog IC design is to select a circuit topology and determine sizes for its devices. Selection of both topology and device sizes is generally based on models of the circuit's performance, mathematical equations that approximately predict what a given topology can produce and what device sizes are needed to make it do so. These equations are derived from basic electrical laws, the circuit topology, empirical property tables, and so on. Most often, solution of the unknown design parameters from the known performance requirements using these equations is very difficult, and often must be done numerically. Hence traditionally, i.e., when performed manually, at least in the initial stages of design, the equations used are very simple. This makes them easier to analyze, solve and compute, operations which must often be performed over and over before suitable choices for the sought after design parameters are found. Unfortunately, their simplicity also makes them very approximate. Once a tentative design selection has been made, its correctness must still be verified through more thorough analysis.

This is usually accomplished through SPICE-level computer simulation using more precise models than those employed for initial topology and size selection. These simulations generally reveal discrepancies in the predicted and therefore desired performance. When discrepancies are found, corrections are generally required for previous design assumptions, modifications must be made in previous choices, and further verification through simulation is necessary. This selection-simulation-modification design loop may iterate several times before an acceptable design is found. Moreover, as SPICE-

level simulations can be very time consuming, even the models they too employ tend to be simpler in the earlier iterations. However, these models generally become more refined and more accurate as design progresses. In particular, initial simulations will not often include layout effects while later ones may. Later mathematical equations used to select topologies and device sizes may also include layout effects as well, but currently this is less common. Note that to include precise layout effects requires that a design have a layout.

In this way layout synthesis may find itself within an overall design optimization loop which includes circuit topology selection and device sizing. The various iterative optimization strategies just described are illustrated in figure 12.

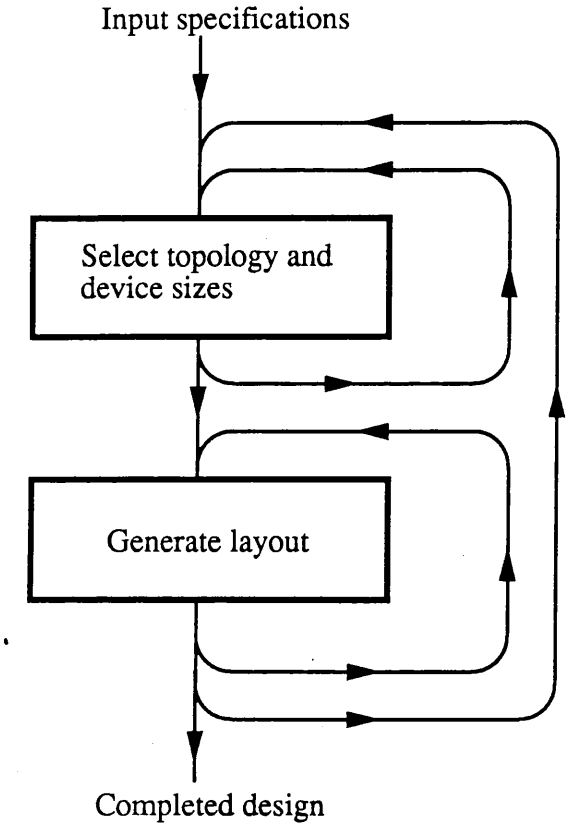


**Figure 12:** Possible optimization loops for analog IC design.

It should be pointed out that the outer most loop in this figure, the one including circuit selection and layout, need not optimize both simultaneously. It may exclude either topology considerations, somewhat as Hong and Allen and Gyurcsik et al. have done, or it may exclude geometry effects, so as to alter only topology and device sizes.

The latter loop optimization possibility is in fact the more common. Within the analog IC design process overall, with or without layout effects taken into consideration, topology selection and proper device sizing are generally considered to be the primary design bottleneck. For this reason, much effort at automating analog IC design has been directed at this stage of the design process. An influential early attempt at this was that of DELIGHT.SPICE.[56, 57] This system coupled a general purpose design optimization tool, DELIGHT, to the standard IC circuit simulator, SPICE. The SPICE program acted as a function evaluation routine, providing circuit performance measurements to the optimizer, DELIGHT. The optimizer utilized this information to perform a numerical optimization on various continuous circuit parameters, typically device sizes, selected by the user. The optimization criteria, also provided by the user, could involve any performance measures determinable by the SPICE program. Other systems using this same technique but varying in such details as the type of simulation or optimization method employed, have since been reported as well.[48, 76] As a direct, brute force automation of traditional precision optimization methods for device sizing, this technique can be effective, but it is slow by automation standards due to the often enormous number of simulations needed. This slowness in fact typically has limited the number of design parameters that can be simultaneously optimized with the approach to around ten or twenty.

More recent efforts at automatic circuit topology selection and device sizing that use iterative optimization methods have attempted to avoid or reduce the heavy computational cost of detailed simulations by focusing on more effective prediction of circuit performance through more direct mathematical circuit characterizations. OPASYN[44] and IDAC[19, 23] for example, are two automatic analog synthesis systems that take such an approach for device sizing. Each of these systems selects device sizes through a numerical optimization procedure. The mathematical circuit models employed by these procedures are more accurate, however, than those typically used in manually directed computer or hand calculations. By basing device size selections on more accurate estimates, the initial guesses tested through simulation can be closer, subsequent modifications can be made more intelligently, and consequently, the number of precision SPICE iterations necessary before a suitable solution is found can be reduced. The equations employed by Koh et al. include parameters that allow them to be fitted

empirically to past simulation results so as to better predict similar future ones. In these systems, these equations are either part of a template (OPASYN) or procedurally encoded (IDAC), and thus must still be manually derived and entered.

Barlow et al.[8, 9] take this idea a step further by automating the equation derivation process as well. That is, their system analyses the circuit topology directly, automatically constructing its own symbolic mathematical expressions needed to characterize the topology's behavior for device sizing operations. Not only does automatic model derivation immediately make their system more general, but it also allows them to employ models too complex for human manipulation. These models can therefore be more accurate. They report that voltage and current predictions made using these automatically derived equations generally match the predictions of SPICE to from four to five decimal places.

Because it is an extensive area of research activity, much more could be presented on the techniques that have been tried for automatically optimizing analog IC design features that are not directly related to layout. (Readers so interested in this subject might consider (19).) However, this is not our main objective. With regards to layout synthesis, it only remains to point out that topology selection and device sizing carried out in the abstract, that is, without concern for the physical characteristics of these features, is obviously simpler than choosing them as more concrete attributes of a layout. The drawback is that doing so does not allow physical effects to feed back properly to influence the search or optimization process.

To get layout effects fully into the optimization loop, it is possible as Onodera et al.[65, 66] have done, to extend the automatic device sizing technique of DELIGHT.SPICE by brute force. They do this by first describing the layout procedurally, including in this definition the calculation of all important parasitic parameters along with the calculations for the geometry. This permits all layout induced electrical effects of interest to be determined relatively quickly. Thus, given a set of candidate device sizes produced by a numerical optimizer acting as system driver, a complete set of layout parasitics is obtained by executing the layout generator, and included in the characterizing SPICE simulations. The performance measures fed back to the optimizer by the simulator thus include these effects. For most circuits, the additional time required to run the layout generator for parasitic calculation will be insignificant in comparison to the computational cost of the subsequent simulation. This is only true, however, because the simulation costs are so high. Moreover, manual development of the procedural layout generator, with its precise parasitic calculations to boot, in general, will be very tedious, time consuming and difficult.

To avoid the need for extensive simulations, techniques based on more accurate direct performance prediction similar to those developed for layout-independent optimization can be proposed. Likewise, automatic equation derivation can be suggested to alleviate the need for tedious manual encoding of parasitic calculations. When layout effects are included in either of these schemes however, they become much more difficult to implement. Both of these capabilities are currently at the frontier of research and development in integrated circuit computer aided design. In fact, for analog design, only reports somewhat peripheral to these objectives have been published to date.

Smith et al.[78, 79] for example, circumvent most of the difficulties involved with incorporating layout effects into overall design optimization by imposing a simple structure on the circuit layout. A cell in this system consists of a single, fixed-height strip and a single dedicated routing channel. A circuit topology is laid out by arranging fixed and parameterized device cells in the strip, and connecting them within the dedicated channel. This simple structure allows circuit models containing device sizes as parameters to include layout derived parasitics relatively easily. Layout and parameterized model extraction can both be done automatically. In this system, given a set of device sizes, parasitics are directly calculated and used to predict circuit performance either through direct estimation equations or through simulation. Although simple and fast, this scheme sacrifices a great deal of layout freedom and hence some performance as well.

How this shortcoming might be removed is suggested by a system developed by Obermeier et al.,[58, 60, 61] called EPOXY, for automatic transistor sizing in digital ICs. In this system, geometries for sizable devices, which in digital ICs are always transistors, are placed on movable horizontal and vertical *virtual grids*. The spacing between the grids is adjustable so that as device sizes grow or shrink, the layout as a whole can be grown or shrunk, as need be, by moving the virtual grids either to maintain adequate separations or to close up otherwise unused areas. Interconnection wire lengths are also grown or shrunk when the grids are moved so as to maintain proper connectivity. Given a layout description in terms of virtual grids such as this, it is possible to derive equations relating interconnect lengths to device sizes. EPOXY derives these equa-

tions automatically by analyzing a virtual grid description of the layout.

With such equations at hand, it then becomes possible to incorporate them into further expressions for predicting performance. EPOXY in fact derives these equations automatically as well, and then utilizes them in a numerical optimization algorithm to choose transistor sizes according to a user specified optimization criteria. To date, however, this system has only been applied to digital circuit optimization, which is different than analog optimization in several ways. For one, the usual optimization criteria for digital design involve primarily timing delays and power consumption (in addition to total area). These criteria have relatively simple relationships to geometry as compared to many effects of concern in analog design. This makes them easier to derive automatically. Secondly, no layout constraints, such as symmetry, are accommodated by the system. Despite these inappropriate qualities, the approach and the architecture of the system have much to suggest their possible adaptation for analog work. The most important of these is the very general virtual grid layout model, which permits significantly more design freedom than does the single strip architecture assumed by Smith et al.

Another of the system's features worthy of adaptation for analog design is its open architecture. (See figure 13.) As implemented, their is much independence between system components. The equations relating performance to geometric measurements, for example, are specified independently, and can be supplied or modified by the user, as can the numerical optimization algorithm employed. In fact, the system is actually intended to provide a general *framework* for transistor sizing rather than a single specific technique for doing so. The value of providing an open framework in which different procedures, optimization criteria, design equations and so on can be applied to different problems is greater for analog design than for digital. This is because, in analog design, the issues that can be of concern and the techniques for addressing them are more varied. For analog design even more so than for digital, there is no universal design model or solution strategy that will work for all situations. Hence, accommodating flexibility is an important first step towards realizing more general, more widely applicable tools.

Recognizing the value of an open architecture for automatic analog synthesis, Berkcan and d'Abreu[10, 11] have proposed an overall framework, aimed specifically at analog design, for organizing the entire synthesis process, from topology selection

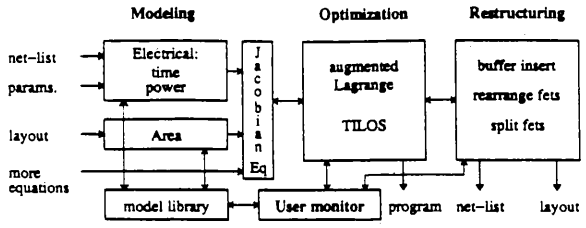


**Figure 13:** Modular structure of EPOXY (from (60)).

through layout, that can coordinate any and all types of design optimization commonly used. This framework is based on top down design through successive decomposition, augmented by feedback from below in the form of verifying simulations. The overall design flow utilized is depicted in figure 14.

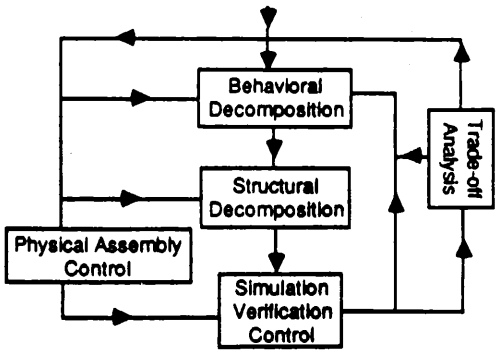


**Figure 14:** Design flow for a complete, automatic analog IC design system (from (11)).

In this system, given a set of performance specifications, the design process begins with the invocation of an application-specific search mechanism, which selects a tentative template. In addition to a circuit topology, each template includes attached procedures for translating the specifications for the circuit as a whole into requirements for its subcomponents. If the subcomponents are devices, determining requirements for these subcomponents amounts to selecting their sizes. However, a topology entered in a template can also be somewhat more abstract, containing complex subcomponents, as well as discrete devices, that must themselves be designed. The translation procedures must then generate a new set of specifications for these subcomponents. How these procedures accomplish this translation is not restricted. They may, for instance, employ either direct calculations, an optimization strategy involving simulation, or any combination

thereof. Once a given circuit's specifications have been translated into those for its subcomponents, the subcomponents themselves, if they are complex, are then decomposed and designed in a similar fashion. This is the mechanism of successive decomposition.

To tentatively verify the correctness of a given set of subcomponent performance specifications, before decomposition has progressed to completion, abstract models, called *macromodels*, are used to simulate the design at its current level of detail. These macromodels, like the requirement translation procedures, are provided by the template. As decomposition proceeds, the more detailed models attached to the templates of the subcomponents are substituted for the higher level macromodels, and the circuit is reverified. New circuit topologies, subcomponent specifications, or templates may then be selected, as needed, to correct identified discrepancies. A design is completed when decomposition has taken a topology completely down to procedurally generatable components (typically individual devices), and a precision simulation of the resulting detailed model verifies that the selections meet the original top level specifications. Because the templates include geometric layout information, as subcomponents and devices are selected, the design is also laid out. Hence, circuit models may include layout specific effects as a matter of course. (Presumably, layout effects may even be included in approximate form in higher level macromodels.)

The significant features of this automated IC design scheme are its use of a hierarchical circuit and layout description model, permitting design by successive decomposition, its use of macromodelling to verify and/or optimally choose subcomponent specifications before these components have been fully described, its use of resimulation upon further decomposition as a general feedback mechanism to verify design choices, and its ability to accommodate multiple design strategies by allowing different procedures and macromodels to be associated with each template. The use of macromodeling and incremental decomposition when verifying designs through simulation make this scheme more efficient than others that rely entirely on full detailed simulation at every stage. Because it can include topology selection in the design/optimization process it is also more general than techniques based on numerical optimization that can only select continuously variable parameter values. (Obermeier's system can also perform topology selection,[59, 61] but the method employed is much more ad hoc.) Moreover, by attaching procedures and models to templates, the system achieves a high level of flexibility and openness. In addition to these architectural advantages, the layout specifications attached to each template are constraint-based, allowing full generality in layout geometry, as well as in modeling its effects.

The primary disadvantage of the approach over those previously described is its much increased complexity. The proper functioning of a system structured in this way would require intelligent partitioning of circuit topologies into subcomponents, including in this partitioning adequate classifications as to what subcomponent variants (templates) are to be substituted under what circumstances during decomposition. Each component would most likely need a good number of carefully and individually configured templates if high performance designs are to be expected from the system. Moreover, the procedures that translate circuit specifications into subcomponent requirements, and that select circuit topologies must still be developed manually. These complexities are essentially the price paid for the greater generality of this approach over more complete but more specific alternatives. Nevertheless, due to its flexibility, extensibility and openness, adherence to a general automation framework such as this one might substantially reduce the difficulty and time required to develop truly extensive and capable automation tools for analog design than might reliance on purely ad hoc procedural system development methods.

## 4. Analysis and Conclusion

### 4.1. Summary

Due to its reduced complexity, automatic cell assembly of high level functional blocks, including general floorplanning, the topic of section 3.1, is the most mature of applications for automatic analog IC synthesis. Using clustering techniques for placement, and routers specialized for analog circuitry, it is now possible to perform analog cell placement and routing with acceptable quality as automatically as these tasks can be done for digital circuitry. As one progresses down the complexity curve towards the smaller but more delicately designed circuits, however, the gap between levels of analog and digital design automation widens. Tools do exist for automatic synthesis of analog circuits at all circuit complexity levels, but those for the most demanding applications, e.g. single op amp synthesis, are currently either very specialized, or very experimental. For example, the parameterized op amp cell generators of the Concorde system can produce near manual quality layout, but only over certain ranges

of the parameters, for specific process technologies, and only for op amps. The much more general ANAGRAM II[22] system on the other hand, can produce layouts for many extremely varied circuits, but not as yet very competitively.

With the exception of ILAC, which is somewhat more advanced, current state of the art design automation systems in commercial use for automatic analog IC layout rely either on the completely procedural "silicon compiler" approach, or on what might be called a "standard generators" framework, in which only the most difficult module internals are laid out by single purpose, specialized procedural generators, while their overall placement and routing is handled by more general tools in a manner similar to automatic standard cell assembly. The techniques on the research horizon that are most highly developed are those relying on some form of static, declarative specification format that is still very general and very expressive. These would be the techniques utilizing constraint-based methods or other forms of flexible templating. There does not appear to be any good reason why systems utilizing these techniques could not produce competitive designs just as well as procedural methods. When fully developed, moreover, they could provide substantially greater flexibility and ease of development than current procedural techniques. This state of affairs, the current state of the art together with solid indications of improvements in the future, indicates that automatic analog IC layout synthesis has, if anything, proved itself to be doable.

Besides noting just *what* has been accomplished, another important observation that should not be overlooked in assessing recent progress in analog IC automation is *how* these accomplishments have been achieved. Two general strategies seem to have predominated in the successes over a much less conspicuous third.

The first strategy is what might be called "generalization through massive specialization," or "detail-intensive." The idea behind this technique is to ensure adequate design success by providing an enormous variety of special purpose design alternatives each of which is individually assured of functioning well over its own specialized range of applicability. The reliance on vast amounts of special case details can be easily discerned in the use of multiple, special purpose, individually developed module generators. It is especially apparent in the use of templates, whose purpose, similarly, is precisely to make it easy to represent and retrieve large quantities of narrowly applicable plans, procedures, design details, or what have you. Fundamentally, the detail-intensive approach is based on human guidance in the design process. It is essentially a generalization of the idea of a standard cell library. Within an automated system structured according to a detail-intensive paradigm, most of the skill necessary to design a module ultimately originates from manually made decisions, just as it does when manually developing a cell for a standard cell library. The primary difference is that the input directly provided by the human is represented more generally, so that it may be applied more flexibly. Had not so many important overall circuit performance measures depended so critically on such low level particulars as the sizes of individual devices, the use of fixed standard cells might have sufficed for analog design as it has for digital. This might have obviated the need for such things as module generators and templates, but such has not proved to be the case.

The second general strategy often employed by successful automatic analog IC design systems is massive application of computer power, particularly in the form of extensive design space searching, and extensive, precision design assessment through brute force inclusion of every design detail. This approach can be seen in the use of such computation intensive search methods as simulated annealing and numerical optimization techniques, and in the extensive use of SPICE-level simulations for circuit characterization. Unlike the detail-intensive approach, the human input is negligible in these systems, being confined for the most part to proper problem statement. The path to the solution finally obtained is typically neither observed by nor even explainable to humans. Thus the principle behind this approach is that a good design solution can be found and identified from its performance characteristics alone, and need not involve human understanding. Application of this strategy often also tacitly implies that good circuits can *only* be found through such methods.

Both the detail-intensive strategy and the compute-intensive approach have quite undesirable drawbacks. Detail-intensive methods require that much human effort be expended in order for them to work. Compute-intensive methods, on the other hand, are often extremely time consuming. The need to use one or the other of these methods has apparently been motivated on one hand, by the many widely varying, unpredictable characteristics of many analog IC design features, such as the large ranges of device sizes that can occur, and on another hand, by the non-systematic nature of many analog IC design tasks, such as device sizing. It should be noted that these qualities are inherent to analog IC design in general, and are not simply a property of

its automation.

The third and least used strategy for automation of analog IC layout is the application of systematic methods based on simple principles or heuristics. These would include methods that applied such techniques as linear programming, or some variant of depth-first search. To be sure, examples of this strategy can be seen in the clever techniques for analog channel routing described in section 3.1, such as that of Gyrucsik et al.,[30] or the stylized floorplans employed for switched capacitor circuit layout, but by and large, these techniques, when applied to analog layout, have generally resulted in systems of either little generality or that produce circuits with substantially compromised performance.

In contrast, principled and heuristic methods actually predominate in digital IC design automation. The prevalence of clever heuristic techniques such as the left-edge algorithm for routing, or min-cut for floorplanning in digital layout automation can be attributed to the relative simplicity of digital layout as compared to analog. It is not as hard in digital layout to ensure proper performance, as fewer things can go wrong. This makes a primary determiner of quality for an automatic digital layout system its speed of execution. Speed of execution quite often is achieved through clever programming. Fast heuristic algorithms however, tend to abstract their solution space into something too simple to represent all the varied interactions and concerns in an analog layout. Hence, when they are applied to analog layout they tend to neglect important concerns and yield inadequate results in many cases.

### 4.2. Analysis

Given that automatic layout of analog ICs is indeed feasible, with general strategies available for going about the process, why is the state of the art in automatic analog layout and design not as advanced as it is for digital?

For one, circuits for analog applications, as has been noted, tend to be rather simple. Their components, when integrated, moreover, tend to be rather large, with sizes being less determined by minimum lithographic resolution as with other characteristics of the IC fabrication process, characteristics that have not changed as much as the technology has improved. Hence, even today, levels of integration of analog ICs and analog subsystems within analog-digital ICs remain at device counts that in the digital world would be classified as falling in the MSI or LSI range, rather than the VLSI range, but it is only within the VLSI domain that automatic synthesis tools truly become essential. Moreover, one cannot neglect the fact that until only recently has there been much demand for analog IC design automation. Only since the emergence of practical "systems on a chip," containing both analog and digital circuitry, has there really been a substantial need for new analog designs. The small demand meant few analog IC designers and a smaller, less attractive market for IC CAD tool developers. These two factors, simplicity of circuitry and low demand, can account for the slow pace of analog IC design automation up until a few years ago.

However, now that demand has picked up, why has computer aided design of analog IC layout not caught up as well? Given what has been accomplished in the field of analog IC design automation to date, the reason for this is clear. It is because analog IC design automation is genuinely different from that for digital IC design. It is enough so, at least, that it has not been possible to catch up simply by adapting tried and proven digital automation methods for the purpose.

A fairly pointed and representative example of how digital and analog IC layout are so different can be observed by considering their respective treatments of routing. The primary goals in digital layout in general are minimization of area and minimization of interconnect delays. These are both furthered by minimization of wire lengths. In digital layout, area and wire length minimization are thus typically the only objectives of both floorplanning and routing. This allows floorplanning to proceed independently of routing through the use of simple but sufficiently accurate wire length estimators. Actual detailed routing can then be performed afterwards, accepting the floorplan as a given. In contrast, for analog IC layout, wire length can be so low in priority, as compared to such concerns as crosstalk and matching, that it may be virtually ignored. The more important concerns in analog routing, moreover, tend to be highly dependent on relative component placement. A crossover, for example, might be inevitable with some placements, but avoidable in others. This means that routing cannot accept a floorplan as "a given," and cannot proceed independently of floorplanning.

In analog IC layout, idealy, routing would precede placement, were this possible. In practice, as we have seen, routing has indeed determined placement to a much greater extent than is the case for digital layout. The connection-driven clustering techniques of Mehranfar,[52] of Kimble et al.,[42] and of

Winner et al.[85] are good examples of this. Also, the way in which Yaghutiel et al.[87] and Barlow et al.[9] allow their routing algorithms to detail, fix, or correct initial placements clearly indicates, for analog layout, the dominance of routing concerns over simple area-minimization-directed floorplanning. Whether an analog IC is manually or automatically laid out, routing is always a primary concern. It directs the entire layout process, *driving* floorplanning rather than being driven by it. In analog IC layout, it is the floorplanner that must often take certain routing restrictions as givens rather than the router that is constrained.

Routing, however, is but one example of the differences between analog and digital design. From a more general perspective, analog IC layout is seen to be different from that of digital in being more complex, and less well understood. As we have also seen, analog IC design, including layout, is based to a much greater extend than digital design on empirical experimentation. Thus, the methods of analog IC design tend to be the methods of efficient exploration, where as those of digital design tend to be more systematic, algorithmic and principled. It is not possible in analog IC layout, for example, to employ a systematic procedure like the min-cut algorithm to floorplanning, because the modules to be arranged vary widely in size, and such an algorithm could not account for routing concerns like crosstalk. Device sizing is hard to efficiently systemize because there is no guiding theory comparable in power to boolean logic as there is for digital circuitry. For similar reasons there is no more a method for systematic analog IC layout as there is one for systematic analog IC device sizing. Although there may indeed someday emerge systematic methods for analog layout and for analog IC design in general, the systematic methods that exist today, for digital design, have not proved applicable to analog work. Thus we find analog IC design automation, to a great extent, now evolving on its own, and at no greater pace than that of digital.

Be this as it may, what then are the fundamental problems with which current evolution in automatic analog IC layout is now challenged? Of course, there is always the need for incremental improvements in the capabilities already proven. Few if any automatic synthesis tools in use today, whether for analog or digital layout, produce results as good as those that could be obtained, at least in principle, by intensive manual methods. Thus the increased design efficiency these tools provide is bought to some extent at the price of lower overall design quality. This indicates that there is still room for improvement even where automated technologies have already shown themselves worthwhile. Automation of analog IC layout, however, still has several much more pressing challenges than design quality. These more important automation problems derive, for the most part, from challenges inherent to analog IC design in general.

Note that one of the reasons analog IC design is so empirical is that many of the physical phenomena affecting analog IC electrical behavior are too poorly understood. Here the problem is not so much that undiscovered laws of physics may reveal themselves in unexpected circuit behavior so much as it is simply anticipating those phenomena that are known but not recognized as significant. Thus, in analog IC design today, it is still possible to discover after the fact, for example, that thermal variations needed to be approximated to second order in a new design, even though in several ostensibly similar previous designs, first order approximations had worked quite well. In analog design, such unanticipated needs can lead to catastrophic failure. That is, not only might the circuit not meet specification, but it might not even come close, or function at all. Although dealing with such adversities is an analog IC design expert's bread and butter, his raison d'etre, there are currently no known ways of handling them in an automated manner. That is, knowing what to model and when to do so is still very much a part of analog IC design and an activity that is currently much too poorly understood to be automated. Fortunately, useful automatic analog synthesis tools can still be developed without this capability if their limitations in this regard are well characterized. However, even characterizing exactly when the models assumed by a given automation system are likely to be valid poses considerable challenges of its own.

Another, less dramatic but more pervasive, problem facing analog IC layout and design is that, even when well identified and properly anticipated, the physical and electrical effects needed to properly model the operation of an analog IC are too complicated to be easily understood or manipulated. This is the reason analog IC design relies so heavily on simpler, easier to understand, *approximate* design equations, and on precise, computer generated SPICE-level circuit simulation. It is also the reason why automatic circuit model generation and manipulation have recently proved useful. The need to perform extensive and tedious manipulations and calculations on complex and cumbersome circuit models and performance equations has been one of the major bottlenecks in analog design from its inception, and

this is not likely to change soon.

Systemizing such extensive, tedious operations is therefore a principal concern of analog IC design automation initiatives, but formulating automatable design methodologies around such detailed, low level modeling activities leads to yet another design automation challenge. How does one, for example, perform top down design when all the top level performance issues depend so intimately on bottom level particulars, details whose full effects can only be ascertained, with the requisite accuracy, by analyzing (e.g. through extensive SPICE simulations) a completely detailed model of the entire circuit? The use of macromodeling proposed by Berkcan and d'Abreu[10] (section 3.3) and automatic derivation of circuit equations as performed by Barlow et al.[9] and Obermeier[60] (section 3.3) offer general solutions to these problems, but implementing the details of these ideas, that is, developing the necessary macromodels, or efficient automatic design equation solution techniques, are areas of research still quite open and challenging.

Yet another very pervasive challenge in analog IC design automation is providing significant levels of process technology independence. As has been well recognized for many years now in the world of digital IC design automation, a tool that is only useful for circuit design with a given fabrication process quickly becomes obsolete. Generalizing design tools to work with many fabrication processes is now standard practice in digital IC design automation. In digital IC design automation, this generalization is accomplished by parameterizing all process technology dependencies identifiable within the tool. Applying the same technique to analog IC design automation, however, would require much more work, and in some situations might still not be fully effective.

This route to technology independence assumes that a given fabrication process can be fully characterized, for the purpose at hand, by a finite set of process "design rules" and electrical characterization measurements. The problem is that analog ICs depend on considerably more properties of a fabrication process than do digital ICs, and these dependencies within the tool itself will be much more extensive. For example, just knowing the minimum permissible line width and the average resistivity of a given layer of interconnect is usually not sufficient to characterize this layer for analog circuit design, although it most commonly would be for digital work. For the analog circuit designer, it is also necessary to know the *tolerances* on both of these numbers. This is because process manufacturing variations determine many key performance measures. It may even be necessary to measure some process characteristics as a function of other properties, such as temperature or current levels. Moreover, like designing the analog circuit itself, *discovering* which process characteristics are important and must be measured or modeled carefully can depend on both the given process and on the given application.

Even when properly characterized, however, separating a process from a given design procedure, automated or otherwise, can be difficult for analog design. In analog circuitry, seemingly minuscule differences between fabrication process characteristics can give rise to fairly extensive differences in overall circuit design. For example, the degree of isolation or tolerance available on a given interconnect layer can make some circuit topologies or device types suitable at a given level of performance and others not. Due to dependencies such as this, many fabrication process details can become intertwined and insidiously hidden in many design decisions, and thus in the features of the resulting design. Consequently, a given analog IC design, thought to be "fully parameterized" against a given set of process technology descriptors might, for example, still prove intolerably low in performance when adapted for manufacture on a process other than the one used "as an example" in its development.

Both identifying all the fabrication process characteristics that are important to analog design, and factoring them out of the design process, particularly the latter, are two problems that have yet to be addressed thoroughly in analog IC design automation.

In fact, dealing with the problem of technology independence is an issue that has not yet been thoroughly addressed for analog IC design in general, manual or automated. Whether performed manually or with automatic support, most analog IC design work undertaken today is carried out with a given fabrication process in mind, because process independent analog IC design is simply too complex at the present time to be cost effective. Given the possibility of unanticipated circuit or process characterization errors as described previously, it can, at times, also be completely futile. Thus, *accommodating* rather than removing process technology dependencies is perhaps a more rational objective for automatic analog IC design efforts, at least in the near term.

## 4.3. Predictions

### 4.3.1. Short term predictions

From the recent advances and remaining problems that have just been described, it is possible to identify several emerging trends that indicate where the technology and art of automatic analog IC layout and design are likely to go over the next several years.

The most certain of these trends is the expanded use of computation intensive detailed simulation to predict circuit behavior. Two factors are contributing to this movement.

First of all, as automation efficiencies continue to reduce the time spent in pre-fabrication phases of IC design, and a greater proportion of IC development costs shifts to post-initial-fabrication analysis and correction, the economic value of correctly predicting actual chip behavior before fabrication should grow. This can only be accomplished by developing more complex and detailed electrical models, which include more physical effects, and approximate them more accurately. Consequently, time spent performing already indispensable circuit simulations should grow.

The second factor contributing to the greater use of computation intensive simulations is the continuing decrease in computational costs. The computational power of engineering workstations, especially floating point calculation, which is a sizable constituent of analog simulations, has increased dramatically in recent years and continues to rise even as prices fall. A SPICE-level simulation that would have taken several minutes to perform on one of the faster workstations available five years ago can be performed in seconds on systems available today. With execution times dropping into seconds, it becomes practical to use detailed simulation much more freely, for example, in extensive optimization loops, to simulate a larger portion of a design all at once, or to employ much more elaborate circuit models, as just described. Current automated analog IC design methodologies, having evolved over several years when computing resources were not so abundant, do not as yet take full advantage of these capabilities. Thus, future automated analog IC design systems can be expected to make much greater use of detailed simulations as ways are found to exploit current and emerging computational capabilities.

In essence, the current and future cost/performance ratio of computational capabilities available to analog IC design engineers is an unfulfilled opportunity just like any other. Although it may not be clear just exactly how this opportunity will be taken advantage of, it seems certain that it will. Thus, how automated analog IC design methodologies will need to be altered to take advantage of the increasing availability of computational resources, and to deliver more reliable first-silicon designs remains to be seen, but these changes will, in all likelihood, occur.

The next few years will also likely witness an expanded use of detail-intensive automatic design methods, be they procedural specification techniques, use of templates, or some other technique not yet developed.

As high performance analog IC designs are likely to remain highly specific to their given application, and highly technology dependent for some time, it will not be possible to produce such designs automatically except in a technology dependent and application specific way. This, however, will ultimately require the input of many technology and application dependent particulars. The only way to do this, short of developing massive libraries of narrowly applicable analog standard cells, is through something like the development of a library of more flexible generators or templates. Thus, the need should continue for some sort of format that can be used to describe specialized but still flexibly applicable design information. This will necessarily also include a need for general frameworks for manipulating such design information, in whatever form it is represented. This latter need is likely to result in a push towards more open, flexible, customizable frameworks in general.

Fortunately, in this respect, analog IC layout is not much different than the layout of more complex digital circuits. Whether analog or digital, above a certain level of complexity, developing a library of fixed cells becomes impractical. Thus in digital IC design, it has been found useful to generalize the specification of broad classes of complex but still specialized circuits such as adders, multipliers, multiplexers, and so on. As a result of this additional, digital demand, the need for frameworks for representing and manipulating these "semi-specific" designs will be large. This demand should drive development of tools that will make detail-intensive design methods easier and more efficient to follow. More powerful and intuitive layout specification languages are a likely outcome of this drive. Likely to be included among them will be improved procedural as well as constraint-based languages. Digital layout is likely to dominate developments in this direction, at least for the near term, this being the result of the larger demand at the present time for

digital tools. However, as the digital tools become available, they will be adopted and adapted for analog use as well, just as is presently occurring with current procedural layout language tools. Tools developed specifically for analog layout are also likely to emerge as demand for analog IC circuitry picks up.

Perhaps the most promising prospect for future innovation in automatic analog IC layout and design is greater, more intensive application of methods that circumvent time domain simulation through other mathematical approximation and solution techniques based instead on more direct circuit characterizations. This primarily involves the use of more complex and accurate circuit models for topology selection and device sizing. Direct characterization methods are, in general, considerably less computationally demanding than detailed simulation based on temporal integration, even when their solution must be found numerically. Moreover, expressions that directly relate performance to design parameters are usually more useful for optimization and search, as their use is not limited to prediction under a fixed set of initial conditions. Hence, the more effective application of direct relationships between performance and design features holds the potential for substantially reducing computational costs of analog IC design that are now the result of extensive simulation and search.

Traditional direct analog circuit characterization methods, those which can be found in text books on the subject today, require no more automatic computational capabilities than can be provided by a pocket calculator. These techniques have thus not fully exploited computational abilities that are now readily available for analog design. Moreover, because current automatic analog IC design methodologies are still mostly based on these manual characterization methods, these automatic methodologies are only beginning to make full use of the enhanced computational capabilities of fully programmable digital computers as well. At issue is the fact that merely automating traditional manual methods, for example, by employing the same approximate design equations used in hand calculation, does not fully exploit the opportunities for automatic design provided by modern computers. When a computer is to perform the calculations, it becomes practical to assume much more accurate circuit models in these circuit characterizations. Thus, developing and applying more accurate circuit models, that are adapted for automatic rather than manual calculation, presents an opportunity for improving the speed and utility of automatic analog IC design aids that has yet to be fully realized.

The foregoing statement would probably be true even if such more exact design equations were themselves still to be derived and computer encoded entirely through manual means. This however, is not likely to be how the majority of such equations will be obtained. Enlisting the aid of a computer for *symbolic* construction, manipulation and solution of mathematical design equations, as well as their numerical evaluation, promises to alleviate this bottleneck to their effective application as well.

Freed from the pencil and paper methods of manual manipulation, it should be practical to employ considerably more complex expressions in the characterization of analog IC designs. That this is so has already been demonstrated by the recent work of Barlow et al.[9] and of Obermeier et al.,[60] described in section 3.3. The systems they developed used automatic symbolic techniques to obtain equations needed for design optimization directly from either the circuit topology or from the layout. Given this proof of utility, the questions now remaining are, how far can automatic symbolic methods be taken to provide more accurate predictions, and how extensively can they be applied. Given the relative recency of its demonstrated usefulness, it is not likely that current, pioneering efforts in this area have fully exploited the potential for automatic symbolic mathematical manipulation to computer aided analog IC design. It can therefore be expected that further progress will occur in this area in the near future.

Automatic symbolic construction and solution of complex design equations, which are then utilized in the numerical solution of still further quantities of design significance, could go a long way towards fuller utilization of computational aids in the traditionally manual process of analog IC design. It can be noted that progress in this direction does not require the further advance of computational capabilities from where they stand today, but only on the imagination and ingenuity of CAD tool developers to identify and implement the methods that will make such progress possible.

One final speculation, if not as yet a substantiated trend, for the future evolution of automatic analog IC synthesis practices, would be the emergence of truly analog-specific automated system architectures. Much of the effort to automate analog IC layout that has occurred to date has been modeled after or guided by experience in the automation of digital IC layout. This has happened both intentionally and without knowing as developers of

automated design tools, familier with the more mature methods of automatic IC synthesis in the digital domain, attempt to apply their skills to the analog design domain as well. The result has been the utilization of many design automation frameworks and practices that are poorly suited to the task. Perhaps the most glaring example of this misdirection is the early, now mostly forgotten attempts to develop libraries of generally useful analog standard cells. As developers of automated design tools realize what can and cannot be borrowed from digital design automation, such errors in initial direction should find correction. Furthermore, as the demand for custom analog integrated circuitry grows, it will provide greater economic impetus for the development of automatic IC design methods suited primarily for analog circuitry rather than for digital.

To be sure, given current successes, at the higher levels of circuit integration, specifically, what has been called in this paper the cell assembly level, we are not likely to see many analog-specific system architectures. Instead, what is more likely to occur is simply further adaptation of digital methods for analog tasks, and greater accommodation for analog concerns in primarily digital tools. These tools could then be used for either digital or analog layout, and hence combined digital-analog layout. This is not surprising. Even when performed manually, it will be recalled, digital and analog layout concerns tend to converge as the level of integration increases.

At the lower, more diverse level of module generation however, what we are likely to see in the architectures of future analog-specific tools is, first of all, a closer integration of what are in digital layout the two separate tasks of placement and routing. Routing is likely to proceed hand in hand with placement, with the layout process as a whole probably being factored into subtasks in some entirely different fashion. For example, layout might proceed bottom up by cluster group, with the more critical subcircuits being identified, formed into clusters and laid out first. Due to the importance of layout in the design of analog ICs as a whole, we are almost certainly going to see a much closer union of layout with topology selection and device sizing. As each of the other, previously described automation trends evolves, this latter one will become more feasible. Utilization of design equations for device sizing and topology selection that include layout dependent effects, for example, should be made more practical by automated methods for constructing such equations. Thus the functional boundaries that now distinguish the various independent synthesis tools for digital layout are not likely to appear in analog-specific tools, as these boundaries are both ill adapted for this task and the means for their removal are already emerging.

Analog IC design, however, is too complex to be automated as a single, monolithic system. This is even so for just the task of layout. The layout process will still need to be decomposed into relatively independently developed, if not executed subsystems. Predicting what these subsystems might look like is still very speculative, but not without value. To mitigate the drawbacks of technology dependence and other factors that reduce the reusability of much analog IC design work, the use of structured or stylized templates might evolve that would allow rapid modification of previous designs for new purposes by either manual or automated means. If automatic methodologies based on structured templating were in fact to emerge, they would almost certainly give rise to a two- or multi-tiered automation strategy. Within such a system architecture, at the bottom most tier, templates would be applied to produce actual layout, but these and other templates at any tier level could themselves be the product of automatic synthesis by templates in higher level tiers. This could be one way to accommodate the technology and application dependencies of current analog IC design practices, while at the same time progressing toward more general purpose methods for the future.

Although it may not be so clear at the present time just what systems architectures may emerge for analog module generation, one thing, at least, is clear. Without a simpler, more viable alternative, such as exists for digital electronics in the form of standard cells, considerable motivation exists for finding some way around tediously hand crafting each new analog IC layout. This in turn should continue to fuel evolution of automated technologies for analog module layout.

### 4.3.2. Long term predictions

Long term technological forecasting is often best made through comparison and analogy to more mature fields. For analog IC layout, very fortunately, a fairly close correspondence can be drawn between the manufacture of integrated circuits and the field of publishing. The purpose of both these industries is the mass production and wide dissemination of graphic information. For publishing, this information consists of the text, drawings, photographs and so forth that are directly viewed by the human user of a published work. For IC manufacture, paper is preplaced by silicon, ink by monolithic etching, text and graphics by fabrication masks, and

direct inspection by indirect interaction through electronic means, but the fundamental activity, exploiting a physical medium to communicate useful information in a useful way, is the same.

Within this overall correspondence, an identification can be made between textual representation on one hand, and digital computation techniques on the other. At an abstract level, they can both be seen to share a common form and function. Both rely upon a simplified, stylized means of expression to make it easier to describe complex ideas within the limited abilities of their respective media. Textual representation employs a fixed, standardized, discrete alphabet to represent symbolic words and phrases in the same way that digital logic uses high and low voltages to represent signals, numerical quantities, addresses, and so on. Neither the form of symbolic characters, words and phrases, nor the interpretation of the bits in a digital register or signal have any relation to the capabilities or physics of the underlying media. It is this freedom from the particulars of the direct form of representation in both digital logic and textual communication that gives them the power to easily describe complex concepts or systems. Such forms of communication were in fact invented to exploit the possibilities for such expressive freedom.

The correspondence between textual literary communication and digital logic design leaves the complement of each form, graphics in publishing, and *analog* design in IC fabrication, to correspond to each other just as well. Graphical illustration, it can be noted, is more difficult than literary composition, much as analog IC design is more demanding than digital design. Moreover, the greater difficulty of each of these fields arises for the same reason: In both graphical illustration and analog IC design, stylized practices are forsaken so as to more efficiently exploit the full capabilities of the medium. Thus, for example, in graphic illustration, lines are not restricted to first compose characters before expressing a thought, but may roam freely over the page. In analog IC design, voltage levels may similarly roam freely in pursuit of efficient and expressive representation.

The greater expressive freedom obtainable by reversion to the fuller capabilities of a medium do not come without a price, however. The exact size and position of dots in an offset photograph can make the difference between an illustration of a familiar face, and one of a stranger. There is substantially less room in graphic design for manufacturing tolerances. Also, when manually created, every detail of a given illustration must be carefully chosen. For example, unlike textual communication, even the line widths employed can make a difference. Very often a graphic design is created with a particular technology in mind, e.g. photo-offset, lithographic, etc., since the exact rendition produced by that technology can make a difference in the message communicated, as well as in exactly what can be communicated and how this is done. Thus the message is much more closely tied to the technology. As we have seen, when they are compared to their digital counterpart, similar characteristics are true of analog IC layout and design as well.

The history of automation in publishing and in electronic circuit design share commonalities with respect to this correspondence as well. The use of electronic circuitry for analog functions preceded its use digitally, just as the drawing of pictures preceded the use of textual forms. However, in either case, once the stylized forms were established, their widespread application and adoption soon eclipsed those of their earlier, still laboriously constructed counterpart. The reasons for this are clear. Both graphic illustration and analog circuit design remained fairly labor intensive, where as the newer, simpler forms proved themselves more amenable to automation. For example, people generally recognize the development of movable type by Gutenburg as the critical breakthrough that made mass produced book publishing possible, even though hand crafted plates for mass producing books, in the form of wood cuts and etchings, existed for years before the emergence of movable type. Illustrations were the more common subject of hand crafted plates because they represented the more efficient form of communication given the amount of labor involved to create them. There never arose any sort of "movable lines" equivalent of movable type for the automation of printable illustration, however.

The two associations that have now been established, of digital logic with textual communication, on the one hand, and of analog circuitry with graphic illustration, on the other, allow a fairly plausible prediction to be drawn, through analogy, regarding the mutual evolution of digital and analog IC design automation.

We currently stand at a point in the evolution of IC design similar to that of the emergence of movable type for publishing. Digital standard cells are making possible more rapid introduction of digital ICs just as movable type accelerated the use of text. So much so has been this acceleration that there is now a much greater demand for analog circuitry to accompany the digital on analog-digital chips, just as the widespread use of books increased the demand for

illustrations to accompany text!

Given this temporal bearing, the extrapolation of where analog IC design is likely to go is fairly direct. For many years, until the invention of photography, graphic illustration remained labor intensive and the temporal and economic bottleneck in the creation of new publications that required it. If one excludes photography, this is true even today. Skilled graphic artists still take longer to train and are shorter in supply than are the legion sources of textual material. Manually drawn illustrations still generally occupy only a small proportion of most text books for this reason. Nevertheless, this state of affairs is improving, as in fact, it always has. Although the level of skill necessary to compose graphical works has always been high, the quantity of labor required has steadily declined through advances in technology. Thus, chemical etching techniques made it unnecessary to directly scribe an illustration into the plate that would be used for its manufacture, and modern electronic drawing aids have recently circumvented the need for even the pen. Given the close similarities in labor and skill requirements, and in media utilization, that thus exist between graphic illustration and analog IC design, it therefore would be prudent to expect that for the foreseeable future, conceptualization and basic description of analog IC functions will remain more labor intensive than that of digital design, and will continue to require a much closer association with the medium of expression (i.e. the technology).

This prediction has a straightforward corollary with regards to analog IC design automation. Just as completely automated formatting of text has so far preceded completely automated drawing of illustrations, so too will completely automatic layout of digital ICs run ahead of completely automatic analog layout. In all likelihood, some human intervention will be necessary in the layout of analog ICs for quite some time. Rather than going away completely, the human input will simply become less direct, evolving, for example, from detailed geometric specification to perhaps more symbolic techniques that use some type of template, or perhaps to some yet more highly abstracted representation method not yet imagined. Such a progression would parallel that of graphic illustration. The graphic artist today is still much closer to his medium than is the literary author, but he works much more efficiently. Although more tedious than typing, illustration by way of electronic sketch pad on a computer screen is considerably easier today than was hand carving wood cuts in the days of Gutenburg.

Thus the most useful and productive innovations in the automation of analog IC layout that we are likely to see for a while, and in particular, for the near term, will be those that do not remove full design expression capabilities from the hands of "higher design agents," such as experienced human analog designers, but instead, simply make such control easier and more intuitive.

### 4.4. Conclusion

A prevailing view among experienced analog IC designers who know the complexities of their profession but who are unaware of recent progress in computer aided design, is that analog IC design is too difficult to be automated, that it is too much of an art to be systemized. This view is also shared among many daunted digital IC CAD researchers who have only observed the arcane and unusual practices of analog IC design from a digital perspective. The most apparent and perhaps most important conclusion that can be drawn from what has been presented in the previous sections of this paper is that this is in fact not the case. Several of the systems described in this paper are now in commercial use. LTX2[24, 42] (section 3.1), Concorde,[32, 33] (section 3.2.1), and ILAC[72, 73] (section 3.2.3) are some of these that are especially significant due to their substantive generality and the high quality of their output. These and other systems demonstrate that design of useful, competitive analog ICs is not beyond automatic means.

In addition, many of the other more experimental systems that have been described suggest very promising new techniques, such as constraint based specification, and schematic annotation, that seem very likely to prove commercially employable in the near future. Moreover, yet other experimental systems indicate that we are very likely still quite far from realizing the full potential of computer automation for analog design. Thus, although it should not be anticipated that levels of automation in analog IC layout will reach those of digital layout, at least not any time in the near future, we can at least expect a continuing, if slow and evolutionary stream of advances for quite some time.

### Abbreviations used in bibliographic citations

CICC IEEE Custom Integrated Circuits Conference

| | |
|---|---|
| DAC | IEEE Design Automation Conference |
| ICCAD | IEEE International Conference on Computer Aided Design |
| ISCAS | IEEE International Symposium on Circuits and Systems |
| ISSCC | IEEE International Solid State Circuits Conference |
| JSSC | IEEE Journal of Solid State Circuits |
| Proc.IEEE | Proceedings of the I.E.E.E. |
| TCAD | IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems |
| TCAS | IEEE Transactions on Circuits and Systems |

**References**


1. Alexander, M., "A spacial reasoning approach to cell layout generation," *CICC*, pp. 356-359, 1986. (Constraint-based specification method used by Barlow et al. for module description.)
2. Allen, P. E. and H. Navarez-Lozano, "Automated design of MOS op amps," *ISCAS*, pp. 1286-1289, 1983. (Early paper on procedural layout (and design) of analog cells.)
3. Allen, P. E., "A Design Methodology for CMOS Analog Integrated Circuits," *ICCAD*, pp. 217-219, 1983. (Early paper on procedural layout (and design) of analog cells.)
4. Allen, P. E. and et al., "The Use of A CAD Tool to Design Analog Integrated Circuits," *ISCAS*, pp. 1223-1226, 1984. (Early paper on procedural layout (and design) of analog cells.)
5. Allen, P. E. and E. R. Macaluso, "AIDE2: An Automated Analog IC Design System," *CICC*, pp. 498-501, 1985. (Early work employing procedural layout utilizing the C language.)
6. Allstot, D. J. and W. C. Black Jr., "Technological Design Considerations for Monolithic MOS Switched Capacitor Systems," *Proc. IEEE*, vol. 71, no. 8, Aug 1983. (Discusses many of the problems with analog layout and how to get around them.)
7. Assael, J., P. Senn, and M. S. Tawfik, "A switched-capacitor filter silicon compiler," *JSSC*, vol. 23, no. 1, pp. 166-175, Feb. 1988. (A digital automatic design system based on procedural specification is applied to the layout of switched capacitor filters.)
8. Barlow, A., K. Takasuka, Y. Nambu, T Adachi, and J.-i. Konno, "An integrated switched capacitor filter design system," *CICC*, pp. 4.5.1 - 4.5.5, 1989. (Classic three-strip switched capacitor layout architecture. Routing is via hierarchical clustering. Clustering is determined by net sensitivities. Clustering guides cell placement.)
9. Barlow, A. R., K. Takasuka, Y. Nambu, T. Adachi, J. Konno, M. Nishimoto, S. Suzuki, K. Nemoto, and K. Takashima, "An Integrated Switched-Capacitor Signal Processor Design System," *JSSC*, vol. 25, no. 2, pp. 346-352, April 1990. (Classic three-strip switched capacitor layout architecture. Routing is via hierarchical clustering. Clustering is determined by net sensitivities. Clustering guides cell placement.)
10. Berkcan, E. and M. d'Abreu, "Physical Assembly for Analog Compilation of High Voltage ICs," *CICC*, pp. 14.3.1 - 14.3.7, 1988. (Describes a hierarchical template structure for constructing cell placement constraint graphs.)
11. Berkcan, E., M. d'Abreu, and W. Laughton, "Analog compilation based on successive decomposition," *DAC*, pp. 369-375, 1988. (Describes the mechanisms for topology selection and device sizing in a top-down, hierarchical specification design system.)
12. Berkcan, E., C. K. Kim, B. Currin, and M. d'Abreu, "From analog design description to layout: a new approach for analog silicon compilation," *CICC*, pp. 4.4.1 - 4.4.4, 1989. (A complete behavioral-specification-to-layout automatic design system is described. For layout, the classical alternating-rows architecture is employed, as is a stylized floorplan for modules.)
13. Berkcan, E. and Y. Yassa, "Towards Mixed Analog/Digital Design Automation: A Review," *ISCAS*, pp. 809-815, 1990. (Mostly a review of Berkcan's own work.)
14. Berkcan, E. and B. Currin, "Module Compilation for Analog and Mixed Analog Digital Circuits," *ISCAS*, pp. 831-834, 1990. (Constraint based specification employing quadratic programming methods for solution and a special purpose language for input)
15. Bloom, M., "Analog standard cells still more custom than semicustom," *Computer Design*, pp. 28-31, March 15, 1986. (Argues that cells in analog standard cell libraries are not general

purpose, necessitating custom alterations.)

16. Brayton, R. K., G. D. Hachtel, and A. L. Sangiovanni-Vincentelli, "A Survey of Optimization Techniques for Integrated-Circuit Design," *Proc. IEEE*, vol. 69, no. 10, pp. 1334-1362, Oct. 1981. (Numerical optimization algorithms for device sizing, etc.)

17. Camenzind, H., "There's no such thing as analog standard cells," *ASIC Technology & News*, vol. 1, no. 4, p. 4, Aug. 1989. (Argues that cells in analog standard cell libraries are rarely used more than once.)

18. Camenzind, H. R. and A. B. Grebene, "An Outline of Design Techniques for Linear Integrated Circuits," *JSSC*, vol. SC-4, no. 3, pp. 110-122, June 1969. (Early paper describing now standard analog IC design practices.)

19. Carley, L. R. and R. A. Rutenbar, "How to automate analog IC design," *IEEE Spectrum*, vol. 25, pp. 26-30, Aug. 1988. (Short survey of complete automatic design systems for analog ICs.)

20. Chen, H. and E. Kuh, "Glitter: A gridless variable-width channel router," *TCAD*, vol. CAD-5, no. 4, pp. 459-465, Oct. 1986. (Constraint-based channel routing)

21. Choudhury, U. and A. Sangiovanni-Vincentelli, "Constraint generation for routing analog circuits," *DAC*, pp. 561-566, 1990. (Describes a method for translating performance level constraints into constraints on individual net parasitics that are geometricaly interpretable by a router.)

22. Cohn, J. M., D. J. Garrod, R. A. Rutenbar, and L. R. Carley, "New Algorithms for Placement and Routing of Custom Analog Cells in ACACIA," *CICC*, pp. 27.6.1 - 27.6.5, 1990. (Describes ANAGRAM II, which respects symmetry constraints and can take advantage of over-component routing.)

23. DeGrauwe, M. G. R. , O Nys., D. Dijkstra, S. Blitz, B. L. A. G. Goffart, E. A. Vittoz, S. Cserveny, C. Meixenberger, G. van der Stappen, and H. J. Oguey, "IDAC: An Interactive Design Tool for Analog CMOS Circuits," *JSSC*, vol. SC-22, no. 6, pp. 1106-1116, Dec. 1987. (Topology selection and device sizing based on manually derived circuit equations and ad hoc procedural solution techniques.)

24. Dunlop, A. E., G. T. Gross, D. D. Kimble, M. Y. Luong, K. J. Stern, and E. J. Swanson, "Features in LTX2 for Analog Layout," *ISCAS*, 1985. (LTX2 is the standard cell design system utilizing "auto-routed" analog cells.)

25. Eaton, G. V., D. G. Nairn, W. M. Snelgrove, and A. X. Sedrai, "SICOMP: A Silicon Compiler for Switched Capacitor Filters," *ISCAS*, pp. 321-324, 1987. (An example of procedural specification for highly structured and regular layout. Techniques are all digital.)

26. Garrod, D., R. A. Rutenbar, and L. R. Carley, "Automatic Layout of Custom Analog Cells in ANAGRAM," *ICCAD*, 1988. (Placement via simulated annealing. Routing via a best-first selection, partial path extension algorithm.)

27. Gray, P. R. and R. G. Meyer, *Analysis and Design of Analog Integrated Circuits,* John Wiley & Sons, New York, 1984. (Standard graduate text on analog IC design.)

28. Grebene, A. B., *Bipolar and MOS Analog Integrated Circuit Design,* John Wiley and Sons, New York, 1984. (A practitioner's text on analog IC design.)

29. Gregorian, R. and G. C. Temes, *Analog MOS Integrated Circuits for Signal Processing,* Wiley - Interscience, New York, 1986. (Standard text on CMOS switched capacitor filter design.)

30. Gyurcsik, R. S. and J.-C. Jeen, "A Generalized Approach to Routing Mixed Analog and Digital Signal Nets in a Channel," *JSSC*, vol. 24, no. 2, pp. 436-442, April 1989. (Constraint-based channel routing with constraints for analog concerns.)

31. Gyurcsik, R. S., S. T. Cochran, and D. W. Thomas, "Performance Driven Evaluation of Bipolar Analog Layouts," *ISCAS*, pp. 827-830, 1990. (Parasitic and matching sensitivities are identified using SPICE and translated into constraints on component placement and routing.)

32. Helms, W. J. and K. C. Russel, "Switched Capacitor Filter Compiler," *CICC*, pp. 125-128, 1986. (Describes facilities for switched capacitor filter design within the commercially available Concorde silicon compiler system.)

33. Helms, W. J. and B. E. Byrkett, "Compiler Generation of A to D Converters," *CICC*, pp. 161-164, 1987. (Describes facilities for automatic A/D converter generation within the commercially available Concorde silicon compiler system.)

34. Helms, W. J., "Measurement of Switched Capacitor Filters Generated with a Silicon Compiler," *CICC*, pp. 14.2.1 - 14.2.5, 1988.

(Measurements are "not significantly lower than those obtained with custom designs.")

35. Hong, S. K. and P. E. Allen, "Performance Driven Analog Layout Compiler," *ISCAS*, pp. 835-838, 1990. (Net parasitic sensitivities are identified with SPICE and used to assign net classifications.)

36. Hu, T. C. and E. S. Kuh, *VLSI Circuit Layout: Theory and Design,* IEEE Press, New York, 1985.

37. Ito, M. and H. Mori, "ALE: A Layout Generating and Editing System for Analog LSIs," *ISCAS*, pp. 843-846, 1990. (Placement and routing utilizing a sized schematic as a template.)

38. Jeen, J.-C., R. S. Gyurcsik, and W.-T. Liu, "A Two-Layer Channel Routing Algorithm for Mixed Analog Digital Signal Nets," *CICC*, pp. 11.5.1 - 11.5.4, 1988. (Constraint-based channel routing with constraints for analog concerns.)

39. Kayal, M., S. Piquet, M. Declercq, and B. Hochet, "SALIM: A Layout Generation Tool for Analog ICs," *CICC*, pp. 7.5.1 - 7.5.4, 1988. (An expert system supervises conventional placement and routing tools.)

40. Kelson, G., "Design Automation Techniques for Analog VLSI," *VLSI Design*, vol. 6, no. 1, pp. 78-82, Jan. 1985. (Describes several methods for simplified analog IC design, including the standard cell approach.)

41. Kim, C. K., E. Berkcan, B. Currin, and M. d'Abreu, "A New Floorplanning Algorithm for Analog Circuits," *CICC*, pp. 3.2.1 - 3.2.4, 1989. (Assumes the classic analog, alternating-rows strip architecture.)

42. Kimble, C. D., A. E. Dunlop, G. F. Gross, V. L. Hein, M. Y. Luong, K. J. Stern, and E. J. Swanson, "Autorouted Analog VLSI," *CICC*, pp. 72-78, 1985. (The alternating-rows constraint is imposed for analog cells within a standard cell strip architecture floorplan. The router utilizes net classification types and preclustering of analog cells.)

43. Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671-680, May 1983. (Original reference on simulated annealing and still the best introduction.)

44. Koh, H. Y., C. H. Sequin, and P. R. Gray, "Automatic Synthesis of Operational Amplifiers Based on Analytic Circuit Models," *ICCAD*, pp. 502-505, 1987. (Automatic topology selection and device sizing using manually derived circuit equations and other data, all stored in templates.)

45. Kuh, E. S. and T. Ohtsuki, "Recent Advances in VLSI Layout," *Proc. IEEE*, vol. 78, no. 2, pp. 237-263, Feb. 1990. (A recent review of automatic layout containing nearly two hundred references.)

46. Kuhn, J., "Analog Module Generators for Silicon Compilation," *VLSI System Design*, vol. 7, no. 5, pp. 74-80, May 1987. (Describes the application of a special purpose layout language, "C5", based on C, for specification of analog layout.)

47. Laber, C. A., C. F. Rahim, S. F. Dreyer, G. T. Uehara, P. T. Kwok, and P. R. Gray, "Design Considerations for a High-Performance 3-um CMOS Analog Standard Cell Library," *JSSC*, vol. SC-22, no. 2, pp. 181-189, April 1987. (Describes circuit topologies that are relatively process intolerant.)

48. Lai, J. C., J. S. Keung, H. J. Chen, and F. J. Fernandez, "ADOPT - A CAD system for analog circuit design," *CICC*, pp. 3.2.1. - 3.2.4, 1988. (DELIGHT.SPICE type optimizer fashioned out of commercially available design tools.)

49. Leler, W., *Constraint Programming Languages,* Addison-Wesley, Reading Mass., 1988. (Includes a good definition, overview and survey of constraint based specification.)

50. Maeding, D., M. Negahban-Hagh, and B. Klein, "Combining Analog and Digital Using Standard Cells," *CICC*, pp. 491-494, 1985. (Describes an analog standard cell approach.)

51. Massetti, M. A., E. D. Johnson, D. L. Mariano, L. D. Smith, D. R. Willmott, M. R. Gruver, R. L. Hedman, B. R. Owens, and M. J. Russell, "A CMOS-based analog-logic standard cell product family," *CICC*, pp. 24.1.1. - 24.1.6, 1988. (Cell assembly for analog cells in a floorplan for mixed digital-analog standard cells.)

52. Mehranfar, S. W., "STAT: A schematic to artwork translator for custom analog cells," *CICC*, pp. 30.2.1 - 30.2.4, 1990. (Module floorplanning guided by annotated schematics or automatic component clustering)

53. Micheli, G. De and A. Sangiovanni-Vincentelli, *Design Systems for VLSI Circuits,* Martinus Nijhoff Publishers, Dordrecht, The Netherlands, 1987. (An advanced treatise on IC synthesis in

general. One chapter is a fairly extensive survey of physical design. The book emphasizes prodedural methods.)

54. Millman, J. and A. Grabel, *Microelectronics,* McGraw-Hill Book Co., New York, 1987. (Standard text on IC design, both analog and digital.)

55. Mogaki, M., N. Kato, Y. Chikami, N. Yamada, and Y. Kobayashi, "LADIES: An Automatic Layout System for Analog LSI's," *ICCAD*, pp. 450-453, 1989. (Pattern matched template provides constraints for a constraint-based compacter.)

56. Nye, B., A. A. Sangiovanni-Vincentelli, J. Spoto, and A. Tits, "DELIGHT.SPICE: An optimization-based system for the design of integrated circuits," *CICC*, pp. 233-238, 1983. (Device sizing via numerical optimization in which SPICE is used for performance function evaluation.)

57. Nye, W., D. C. Riley, A. Sangiovanni-Vincentelli, and A. L. Tits, "DELIGHT.SPICE: An Optimization-Based System for the Design of Integrated Circuits," *TCAD*, vol. 7, no. 4, pp. 501-519, April 1988. (Device sizing via numerical optimization in which SPICE is used for performance function evaluation.)

58. Obermeier, F. W. and R. H. Katz, "An Electrical Optimizer that Considers Physical Layout," *DAC*, pp. 453-459, 1988. (A system for transistor sizing of digital circuits.)

59. Obermeier, F. W. and R. H. Katz, "Combining Circuit Level Changes with Electrical Optimization," *ICCAD*, pp. 218-221, 1988. (Circuit topology is modified to improve performance in a primarily parametric optimization system.)

60. Obermeier, F. W., "An Open Architecture for Improving VLSI Circuit Performance," UCB/CSD 89/522, Computer Science Division (EECS), Univ. of Calif Berkeley, Berkeley Calif. 94720, August 1989. (A digital system with many good ideas that could be applied to optimize analog ICs.)

61. Obermeier, F. W., "A Model Generation and Compilation System for Improving Electrical Performance," *ISCAS*, pp. 852-855, 1990. (Speed improvements for earlier systems.)

62. Ohtsuki, T., *Layout Design and Verification,* North Holland, Amsterdam, 1986.

63. Okuda, R., T. Sato, H. Onodera, and K. Tamaru, "An Efficient Algorithm for Layout Compaction Problem with Symmetry Constraints," *ICCAD*, pp. 148-151, 1989. (Constraint-based compaction. Moves of matching components are coordinated.)

64. Olmstead, J. A. and S. Vulih, "Noise problems in mixed analog-digital integrated circuits," *CICC*, pp. 659-662, 1987. (Provides a good example of the types of difficulties that can arise at the system level when integrating analog and digital circuitry on the same chip.)

65. Onodera, H., H. Kanbara, and K. Tamaru, "Operational amplifier compilation with performance optimization," *CICC*, pp. 17.4.1 - 17.4.6, 1989. (Procedural specification that includes parasitic calculation. Device sizes are numerically optimized as in DELIGHT.SPICE but include parasitics.)

66. Onodera, H., H. Kanbara, and K. Tamaru, "Operational Amplifier Compilation with Performance Optimization," *JSSC*, vol. 25, no. 2, pp. 466-473, April 1990. (Brute-force numerical device sizing using an numerical optimizer, SPICE for function evaluation, and procedural specification for parasitic extraction.)

67. Otten, R., "Automatic Floor-plan Design," *DAC*, pp. 261-267, 1982.

68. Piquet, S., F. Rahali, M. Kayal, E. Zysman, and H. Declercq, "A new routing method for full custom analog ICs," *CICC*, pp. 27.7.1 - 27.7.4, 1990. (Analog constraints are typed and ranked by type, and then enforced sequencially until only one routing option remains or the constraints are all met.)

69. Pletersek, T., J. Trontelj, L. Trontelj, I. Jones, G. Shenton, and Y. Sun, "Analog LSI design with CMOS standard cells," *CICC*, pp. 479-483, 1985. (This is a precursor to the JSSC paper by the first five authors.)

70. Pletersek, T., J. Trontelj, L. Trontelj, I. Jones, and G. Shenton, "High-Performance Designs with CMOS Analog Standard Cells," *JSSC*, vol. SC-21, no. 2, pp. 215-222, April 1986. (Layout specification is primarily procedural. Templates provide some information to guide routing and for geometries that need no parameterization.)

71. Preas, B. T. and J. M. Lorenzetti, *Physical Design Automation of VLSI Systems,* The Benjamin/Cummings Pub. Co., Menlo Park, Calif., 1988. (A good introduction to layout design automation for digital design.)

72. Rijmenants, J., J. Schwarz, J. Litsios, and R. Zinsyner, "ILAC: An automated layout tool for

analog CMOS circuits," *CICC*, pp. 7.6.1 - 7.6.4, 1988. (Simulated annealing applied to a slicing tree floorplan description for cell placement.)

73. Rijmenants, J., J. B. Litsios, T. R. Schwarz, and M. G. R. Degrauwe, "ILAC: An Automated Layout Tool for Analog CMOS Circuits," *JSSC*, vol. 24, no. 2, pp. 417-425, April 1989. (Simulated annealing applied to a slicing tree floorplan description for cell placement.)

74. Rutenbar, R. A., "Simulated Annealing Algorithms: An Overview," *IEEE Circuits and Devices Magazine*, vol. 5, no. 1, pp. 19-26, Jan. 1989. (Discusses the benefits of, and problems and pitfalls with, applying simulated annealing to IC CAD.)

75. Sanchez-Sinecio, E., J. J. Lee, and P. E. Allen, "An Area Optimized CAD Program for Cascade SC Filter Design," *ICCAD*, pp. 21-23, 1984. (Early paper on procedural layout (and design) of analog cells.)

76. Shyu, J.-M. and A. Sangiovanni-Vincentelli, "ECSTASY: A New Environment for IC Design Optimization," *ICCAD*, pp. 484-487, 1989. (Like DELIGHT.SPICE but more user friendly.)

77. Smith, L. D., H. R. Farmer, M. Kunesh, M. A. Massetti, D. Willmott, R. Hedman, R. Richetta, and R. J. Schmerbeck, "A CMOS-Based Analog Standard Cell Product Family," *JSSC*, vol. 24, no. 2, pp. 370-379, April 1989. (Adaptions of a digital standard cell assembly system for mixed analog-digital cell layout.)

78. Smith, M. J. S., C. Portmann, C. Anagnostopoulos, P. S. Tschang, R. Rao, P. Valdenaire, and H. Ching, "Cell Libraries and Assembly Tools for Analog/Digital CMOS and BiCMOS Application-Specific Integrated Circuit Design," *JSSC*, vol. 24, no. 5, pp. 1419-1432, Oct. 1989. (A restricted strip architecture allows parasitics to be extracted parametrically and included in device sizing operations.)

79. Smith, M. J. S., C. Anagnostopoulos, C. Portmann, R. Rao, P. Valdenaire, and H. Ching, "Construction of Analog Library Cells for Analog/Digital ASICs using Novel Design and Modular Assembly Techniques," *CICC*, pp. 25.6.1 - 25.6.5, 1989. (A restricted strip architecture allows parasitics to be extracted parametrically and included in device sizing operations.)

80. Soukup, J., "Circuit Layout," *Proc. IEEE*, vol. 69, no. 10, pp. 1281-1304, Oct. 1981. (An early, somewhat dated and note particularly comprehensive survey.)

81. Sparkes, R. G. and W. Gross, "Recent Developments and Trends in Bipolar Analog Arrays," *Proc. IEEE*, vol. 75, no. 6, pp. 807-815, June 1987. (Reviews the state of the art in analog arrays circa 1986.)

82. Teel, T. A. and D. A. Wayne, "A Standard Cell Approach to Analog IC Design Utilizing Subthreshold Building Blocks," *CICC*, pp. 484-489, 1985. (Describes a few analog standard cells.)

83. Trontelj, L., J. Trontelj, T. Slivnik Jr., R. Sosic, and D. Strle, "Analog Silicon Compiler for Switched Capacitor Circuits," *ICCAD*, pp. 506-509, 1987. (A mixture of fixed and parameterized cells are placed in a strip architecture.)

84. Tsividis, Y., M. Banu, and J. Khoury, "Continuous-Time MOSFET-C Filters in VLSI," *TCAS*, vol. CAS-33, no. 2, pp. 125-139, Feb. 1986. (Includes a breif comparison of various electronic signal processing techniques: digital, switched capacitor, and various continuous-time analog methods.)

85. Winner, G., T. A. Nguyen, and C. Slemaker, "Analog macrocell assembler," *VLSI System Design*, pp. 68-71, May 4, 1987. (Automatic clustering based on net sensitivities.)

86. Wong, D. F., H. W. Leong, and C. L. Liu, *Simulated Annealing for VLSI Design*, Kluwer Academic Publishers, Norwell Mass., 1988. (Describes (essentially digital) applications of simulated annealing to VLSI design.)

87. Yaghutiel, H., A. Sangiovanni-Vincentelli, and P. R. Gray, "A Methodology for Automated Layout of Switched Capacitor Filters," *ICCAD*, pp. 444-447, 1986. (Classic paper on the subject of its title. The classical three-strip switched capacitor architecture is introduced.)

88. Yaghutiel, H., S. Shen, A. L. Sangiovanni-Vincentilli, and P. R. Gray, "Automatic layout of switched-capacitor filters for custom applications," *ISSCC*, pp. 170-171, 1988. (Describes some design results from a switched capacitor filter compiler.)