Optimization Techniques for High-Performance Digital Circuits

Chandu Visweswariah
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598, U.S.A.
chandu@watson.ibm.com

Abstract

The relentless push for high performance in custom digital circuits has led to renewed emphasis on circuit optimization or tuning. The parameters of the optimization are typically transistor and interconnect sizes. The design metrics are not just delay, transition times, power and area, but also signal integrity and manufacturability. This tutorial paper discusses some of the recently proposed methods of circuit optimization, with an emphasis on practical application and methodology impact.

Circuit optimization techniques fall into three broad categories. The first is dynamic tuning, based on time-domain simulation of the underlying circuit, typically combined with adjoint sensitivity computation. These methods are accurate but require the specification of input signals, and are best applied to small data-flow circuits and "cross-sections" of larger circuits. Efficient sensitivity computation renders feasible the tuning of circuits with a few thousand transistors. Second, static tuners employ static timing analysis to evaluate the performance of the circuit. All paths through the logic are simultaneously tuned, and no input vectors are required. Large control macros are best tuned by these methods. However, in the context of deep submicron custom design, the inaccuracy of the delay models employed by these methods often limits their utility. Aggressive dynamic or static tuning can push a circuit into a precipitous corner of the manufacturing process space, which is a problem addressed by the third class of circuit optimization tools, statistical tuners. Statistical techniques are used to enhance manufacturability or maximize yield.

In addition to surveying the above techniques, topics such as the use of state-of-the-art nonlinear optimization methods and special considerations for interconnect sizing, clock tree optimization and noise-aware tuning will be briefly considered.

1 Introduction

Automatic circuit optimization is an essential part of rapidly, repeatably and robustly designing high-performance circuits. The relentless push for ever higher performance in digital circuits, the need to design circuits of greater complexity, the emphasis on custom design and shrinking product cycles have led to an increased interest in optimization techniques.

Given a logically correct circuit schematic, the circuit tuning problem can be stated as that of optimally assigning sizes to transistors and/or wires. The *performance metrics* are (some subset of) delay, transition time, area, power dissipation, signal integrity, additional timing constraints, layout constraints and manufacturability. Most of these metrics are nonlinear functions of the tunable parameters. Each metric can be presented as either an *objective function* or a *constraint*. The *parameters* of the problem are transistor and wire sizes, and these parameters are usually required to lie within *simple bounds*. Many circuit tuning problems are best stated as *minimax* problems in which the optimizer is required to minimize the maximum of a set of functions. For example, the problem may be stated as minimizing the worst delay across several paths through the logic.

This tutorial paper will address various methods that are used to solve circuit optimization problems. Only continuous optimization problems will be considered in this paper, as opposed to discrete problems (such as choice of gates from a discrete library, reordering of input pins and buffer insertion).

Circuit optimization techniques fall into three broad categories: dynamic tuning (discussed in Section 2), static tuning (Section 3) and statistical tuning (Section 4). Application of state-of-the-art nonlinear optimization methods to circuit tuning is a vast subject in its own right. Section 5 is devoted to some practical considerations in the choice and application of optimization packages. Special topics such as interconnect tuning, optimization of clock distribution networks and noise-aware tuning are briefly mentioned in Section 6. Finally, some promising avenues of future research are enumerated in Section 7.

2 Dynamic tuning

Dynamic tuning[1, 2, 3, 4, 5, 6, 7] implies circuit optimization based on dynamic time-domain simulation of the underlying circuit. The typical flow of a dynamic tuner is shown in Fig. 1. Under the control of the nonlinear optimizer, tunable parameters are set to their initial values and a simulation is performed. The measurements of interest, and the gradients of each measurement with respect to all tunable parameters are fed back to the optimizer. Based on this information, the nonlinear optimization package suggests a new *solution vector*, which is a new assignment of parameter values that is expected to improve the circuit. The iterative process is carried to convergence, or until a user-specified maximum number of iterations is reached. Convergence is typically judged by *sufficient stationarity* combined with *sufficient feasibility*, or a determination of infeasibility. Stationarity implies smallness of the *projected gradient* in the sub-

0-89791-993-9/97 $10.00

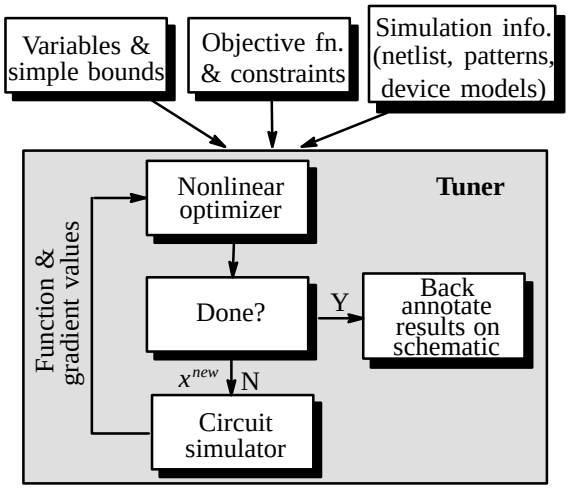

Figure 1: Typical flow of a dynamic tuner.

space of variables that are not at their bounds and feasibility implies that all constraints are satisfied.

The *parameters* in dynamic tuning usually include transistor and wire sizes and they must conform to simple bounds. *Ratio-ing* of transistor widths to one another must be permitted. Further, *grouping* of similar structures is useful to ensure that corresponding transistors of the structures are maintained at the same size during the tuning procedure. For example, a 64-bit multiplexor may have to be tuned so that the corresponding transistors of each group have the same final sizes, thus permitting a regular layout. The *measurements* in dynamic tuning usually include area (often modeled by the sum of the tunable transistor widths), delay and transition time or slew. The objective function and constraints are expressed in terms of these measurements. *Minimax optimization* is a useful feature whereby the worst of a set of measurements is minimized. For example, the problem may be stated as minimizing the worst delay of m paths through the circuit, as shown below.

$$\begin{array}{lll} \text{minimize} & \text{maximum} & c_i(x). \\ x \in \Re^n & i \in \{1, 2, \cdots, m\} & \end{array} \quad (1)$$

Note that the optimizer has no *a priori* knowledge of which of these paths exhibits the worst delay. Further, different paths may be critical during different iterations of the optimization.

The main advantage of dynamic tuning is its accuracy. The tuning is realistic since it is based on full-blown transient simulation. Likewise, *false paths* are avoided in contrast to static tuning methods. If the transistor sizing at any iteration causes failure of a measured signal to switch correctly, the transient simulation is able to detect this situation. In such a case, a *non-working circuit* has been obtained, usually because of the optimizer taking too aggressive a step. Recovery from this situation is implemented by requiring the optimizer to cut back on its step size and trying again.

However, dynamic tuning suffers from a number of disadvantages. The main disadvantage of dynamic tuning is that it is specific to the input pattern sensitizations and measurements specified. Unlike static tuning, it is not possible to tune any but the smallest circuit for all possible input patterns and all possible paths through the logic. The initial parameter assignments should reflect a working and simulatable circuit in which the logical transitions of interest occur.

As with the use of any optimizer, the solution obtained is only as good as the problem specification. Dynamic tuning is particularly vulnerable to designers omitting tacit requirements and then encountering unexpected results. For example, if the input stage of a circuit is tunable, the optimizer may blow up its size, taking advantage of the fact that in the simulation, this gate is driven by a voltage source! Making the input stage non-tunable or constraining the input capacitance seen by the previous stage will lead to more realistic results. Another insidious problem is that transistors that do not switch during the simulation are shrunk to their minimum size in order to satisfy an area constraint, or to reduce the loading on other transistors that do switch. Yet another example is that the target delay on a net may be met by the optimizer but the transition time may be unacceptable. A disciplined approach to accurately expressing all aspects of the problem at hand is essential to making good use of any optimization program!

Dynamic tuning is most often applied to small dataflow circuits in which the critical paths are well known and the input patterns to sensitize these paths are easy to come by. The relative computational inefficiency of these tools also limits the size of circuit that can be tuned.

2.1 Gradient computation

The bottleneck in dynamic tuning is often the computation of time-domain gradients. Gradients are typically computed by the circuit simulator. There are two well known methods of computing the gradients, the *direct method*[8] and the *adjoint method*[9]. The reader is referred to [10] for a tutorial description of the theory behind these two methods. This section will present a quick overview of these two methods and their applicability to the circuit optimization problem.

The direct method is based on direct differentiation of the branch constitutive relations (BCRs) that govern the electrical behavior of the elements of the circuit. The relations thus derived represent a *sensitivity circuit* of the same topology as the original circuit but with different circuit elements. The solution of this related circuit yields the sensitivity of all measurements with respect to a single parameter. Fortunately, the system matrix of the original and sensitivity circuits are the same at each time instant, and hence the cost of LU factorization can be amortized during the analysis of the sensitivity circuit. Note that the sensitivity circuit must be solved as many times as the number of parameters, which is expensive for large numbers of tunable parameters.

The adjoint method is the method of choice for computing gradients of large circuits. In the circuit context, the adjoint method is best understood as an application of Tellegen's theorem[11]. As in the direct method, an associated circuit called the *adjoint circuit* is formed. The adjoint circuit has the same topology as the nominal circuit, but different electrical elements. Like the direct method, the LU factors of the nominal circuit can be re-used during the adjoint analysis, modulo some time point mismatch issues, as discussed below. Control is reversed and time run backwards during the adjoint analysis. Finally, the waveforms of the original and adjoint circuits are convolved to yield the required sensitivities.

The main advantage of the adjoint method is that it yields the gradients of one function with respect to *all* the parameters in a single adjoint analysis. However, because time is run backwards, the nominal and adjoint analyses cannot be carried out simultaneously. Further, it is not easy to make the time points of the two analyses coincide, leading to a clumsy time point mismatch problem. The convolution of waveforms is an additional source of computational and memory overhead in the adjoint method.

The single function which forms the sensitivity function in the adjoint method can be any scalar differentiable function of any number of circuit measurements[6]. Nonlinear optimizers often build *merit functions* which are internal functions that are iteratively minimized to solve the problem at hand. For example, LANCELOT[12] builds the following merit function Φ.

$$\Phi = f(x) + \sum_{i=1}^{n} \lambda_i c_i(x) + \frac{1}{2\mu} \sum_{i=1}^{n} c_i^2(x), \qquad (2)$$

where $f(x)$ is the objective function, $c(x)$ are n equality constraints, λ are the Lagrange multipliers or dual variables and μ is a penalty parameter that controls the weight of the quadratic augmentation of the Lagrangian.

The gradient of such merit functions can be found in the course of a single adjoint analysis. If the optimizer needs only the gradients of the merit function and not the gradients of individual constraints or objective functions, these gradients can be computed extremely efficiently. Even if individual gradients are desired, to the extent that objective functions and constraints are expressed as differentiable functions of multiple measurements, the adjoint method can be applied to compute the gradients of these individual functions as "adjoint groups."

With either method of computing gradients, *chain ruling and combining* of gradients is essential. When the width of a transistor varies, the associated intrinsic parasitic capacitances as well as the diffusion capacitances on the source and drain vary. The sensitivity of each measurement with respect to these parasitics has to be computed, and then chain ruled and combined to obtain the composite sensitivity with respect to all ramifications of the variation of the parameter of interest.

2.2 Practical implementation

Most dynamic tuners are limited to a few 10s of transistors. DELIGHT.SPICE[1] was one of the early practical implementations of a dynamic circuit optimization capability. The ability to handle *semi-infinite constraints* was implemented in that package (see Section 5.2 for a discussion of semi-infinite constraints). Further, the need for a good user interface was stressed.

It has been shown that gradient computation of large circuits is practical provided a fast circuit simulator[13, 10] with device modeling simplifications is used. In this context, gradient computation can be extremely efficient[14, 15, 16, 10, 5, 6]. Depending on the modeling simplifications used, the associated sensitivity or adjoint circuit can be trivial to solve. In the case of SPECS[13], the associated circuit consists of disconnected capacitors, with impulses of charge transferred between the capacitors at times corresponding to event times in the nominal transient solution. Further, the piecewise nature of the waveforms reduces the cost of otherwise costly convolutions. Expressions can be telescoped to create convolution routines that are significantly faster than general-purpose convolution codes.

Recently, a SPECS-based tuner called JiffyTune[5, 6, 7] was reported to tune a circuit with 6,900 transistors in about 2 hours of CPU time (the circuit had 4,128 tunable transistors, 41 delay constraints, an area objective function and involved 125 cycles of simulation). An intuitive graphical user interface was part of JiffyTune, allowing users to specify the tuning problem by pointing and clicking in a schematic environment. Results of tuning were graphically back-annotated onto the schematic. All details of the tuning problem were stored as attributes of the schematic. So if the circuit had to be retuned in any situation (change of technology, change in requirements, re-mapping), the re-tuning could be accomplished at the push of a button. Thus the tuning environment encouraged and facilitated design re-use.

Some remarks about how dynamic tuning fits into the overall design methodology are in order. If dynamic tuning is used on small, full-custom circuits, then the specification of the problem and the layout of the resulting schematic can be performed manually. However, in order to apply dynamic tuning to larger circuits, the difficulty in coming up with input patterns and the concomitant post-tuning layout problem must be addressed. For example, the critical paths as identified at the end of a static tuning (or even a static timing analysis) can be carved out, sensitized and automatically fed to a dynamic tuner. Dynamic tuning is most effective at the schematic level before layout. On an extracted schematic, the correspondence between transistor sizes and associated diffusion capacitances is often lost, making it impossible to obtain electrically accurate gradients. An extractor that produces netlists in which parasitics are parameterized by the size of associated devices would fill this methodology gap which currently prevents post-layout tuning.

3 Static tuning

Static tuning implies circuit optimization based on *static timing analysis*[17]. One of the earliest static tuners was TILOS[18]. In these approaches[19], transistors are usually modeled by equivalent RC circuits. The actual values of the resistances and capacitances are computed during a pre-characterization procedure. The delay of each channel-connected set of transistors is computed using the *Elmore delay model*[20, 21]. Alternatively, delay macromodels are used in [22]. Conventional static timing analysis is used to determine the critical path. The delay of the critical path can then be expressed as a function of the widths of transistors. This expression is a *posynomial function* (a particular algebraic form, see [23]) of transistor widths. The observation is then made that by a simple variable substitution, the posynomial function can be converted to a *convex function*. Thus any local minimum is guaranteed to be a global minimum.

The procedure in TILOS and derivative tools is to start all transistors at their minimum widths, and iteratively bump up the width of the transistor to which the critical path is most sensitive at each step of the algorithm. The procedure is repeated until the lowest critical path delay through the circuit is found. More recently, power optimization has also been proposed in this general framework[24, 25].

The main advantages of static timing analysis are its pattern independence and its speed. Very large circuits can be tuned relatively quickly. All paths are implicitly taken into account because of the underlying static timing basis. The designer is freed of the onus of coming up with input patterns or identifying critical paths. Since many industrial designs are verified by static timing analysis, there are obvious advantages to carrying out static tuning in that same general framework. Further, interconnect delay can easily be modeled and accommodated into this framework.

Unfortunately, static timing analysis has a number of drawbacks. The most serious one is accuracy. Elmore delays do not provide reasonable accuracy in the context of high-performance sub-micron circuits. Thus a gain of 20% or more in performance based on Elmore delay models may not have improved the circuit in reality! Other modeling techniques like "collapsing" each logic gate into an equivalent inverter significantly degrade the accuracy. Unfortunately, the mathematical elegance of mapping the problem into a convex one and the intuitive satisfaction of finding a global minimum are rendered void by the crudeness of the delay approximation. Improving the accuracy of the delay model by, for example, taking into account input waveform shapes, can destroy the posynomial nature of the formulation.

The second major drawback of static tuning is the false path problem. The optimizer may be hard at work tuning false paths through the circuit, and therefore unable to achieve any performance gains in the paths that really matter. But this problem is no more or less serious than the false path problem in static timing analysis, and if the circuit "sign-off" is based on static timing analysis, this activity may be a legitimate one. The third problem with static tuning is the lack of delay models that are functions of transistor sizes. Often, analytic delay models are built up for gates in a library as a function of input slope and output load. These models are quite accurate provided one stays within the "sweet spot," or a reasonable range of input slopes and output loads in which the models are valid. Unfortunately, such models are built by an exhaustive and time-consuming SPICE-based characterization process. These models generally do not exist as a function of transistor widths, thus making them unusable during tuning of transistor sizes. Finally, starting with all transistors set to their minimum size could lead to circuits that may not even have the correct logical transitions. Dynamic tuners, since they are based on a realistic simulation of the circuit, have the advantage of being able to detect such "non-working" circuits and attempting to recover from them.

Static timing analysis in the context of custom circuits is successful only when each *channel-connected component* (or "DC-connected component" or "strongly connected component") is timed using a dynamic simulator of reasonable accuracy under the covers. For tuning purposes, the fast gradient computation methods of Section 2.1 can then be exploited. Unfortunately, there are no credible, gradient-based static tuners built with such timers used for the underlying analysis.

4 Statistical tuning

Yield loss on a fabrication line can be attributed to *catastrophic* and *parametric* (or *circuit-limited*) yield loss. Catastrophic yield loss is due, for example, to dust particles that cause opens or shorts on metal lines. Parametric yield loss, which is discussed in this section, occurs due to inherent manufacturing variations, leading to chips that do not have the required performance characteristics. In *sorted designs* like microprocessor chips, this degradation can mean that insufficient chips end up in the high-performance, high-profit bin. In non-sorted designs (for example, a bus controller chip) circuits below a performance threshold must be thrown away. Across the chip linewidth variations (ACLVs) constitute the single dominant set of parameters that lead to variations in the performance of the circuit. Statistical tuning is the process of changing design parameters to minimize the circuit-limited yield loss.

Aggressive tuning of a circuit often drives it into a corner of the process space, thus causing its yield to suffer. This problem has been studied extensively in the literature and [26] is a good tutorial introduction to the subject. In addition, the books [27, 28] provide a survey of the state-of-the-art and provide extensive pointers to further reading, while [29] is a useful reference on the topic of creating and building statistical models.

The approaches to various aspects of statistical tuning are listed below.

- In *Monte Carlo* analysis, the parametric space is sampled and the design simulated at each sample point. Of course, this method assumes that distributions of the parameters are known. Further, by various *principal component* and *correlation* analyses, the number of independent parameters is reduced so as to limit the dimensionality of the space being sampled and therefore the number of simulations required. The results of the simulation runs can be used to determine both the distribution and worst-case behavior of the circuit.

  Designs are often simulated at multiple *process corners*, which is a simple form of Monte Carlo analysis. It is possible in the context of a dynamic tuner to replicate the nominal objective function(s) and constraints across all process corners, and simultaneously tune at all process corners[6]. Nominal objective functions are transformed into minimax functions across the process corners.

- *Extreme case analysis* is aimed at finding the worst-case behavior of the circuit given a statistical model of the parameter variations. The goal is not to predict the statistical distribution of the performance, but to predict the worst-case. A simple statement of the problem would be, for example, to maximize the delay of a circuit by optimally assigning transistor lengths from a pre-characterized distribution.

- *Yield prediction and optimization* seek to explicitly model the yield characteristics of a circuit as a response surface. Once this is done, the actual parametric yield of a circuit in the face of manufacturing variations can be predicted. Further, based on the yield model, the circuit can be modified to maximize the yield.

- *Design centering* methods do not explicitly compute or model yields. Instead, they take the approach that pushing the circuit deeper into the interior of the feasible region in the space of parameter variations will result in a more robust circuit and therefore higher yields. While design centering methods operate on such a geometric model of the feasible region, method-of-moments-based techniques implicitly attempt to move designs away from regions of low yield to regions of high yield without seeking to explicitly compute the feasible region[30].

Despite much research on the topic of statistical tuning, Monte Carlo and extreme case analyses are the most popular approaches; industrial practice consists predominantly of these two methods. The advantages of these methods that are not shared by the other techniques are that they are easy to understand and in a form that is easily accessible to the design engineer.

5 Optimization methods

5.1 Choice of optimization method

The choice of the optimization method for circuit tuning is crucial. Some introductory comments on this topic are in order. Circuit tuning is a nonlinear problem that cannot be solved by a linear optimizer; no amount of familiarity with linear optimization on the part of the developer can overcome this limitation! From the point of view of mathematical optimization, geometric programming[31] is an old technique whose development predates the considerable advances in the field. Because of its inherent inflexibility it is rarely the most appropriate method and is essentially abandoned by researchers in nonlinear optimization. Our desire to preserve flexibility and solve large tuning problems logically drives us towards the choice of large-scale, general-purpose nonlinear optimization packages.

Circuit tuning is best approached by *gradient-based nonlinear optimization.* In the absence of gradients, large problems cannot be solved and one is typically limited to problems in a few 10s of variables. Worse, there is often no guarantee of convergence or optimality in such "gradient-free" techniques. The efficient computation of gradients and even Hessians (matrix of second partial derivatives) is key to effective optimization of large circuits. Note that gradient-based optimizers attempt to converge to a local feasible and stationary point; there is no guarantee of global optimality.

When tuning is applied to real-life situations, one finds that there is a need for flexibility in the statement of the problem. For combinational logic, we might want to minimize area subject to a delay constraint. For custom, high-performance circuitry, we may want to minimize delay at any cost. For dynamic logic, we may have numerous timing constraints. To handle multiple paths through the logic, minimax optimization is indispensable. Therefore, tuning tools should allow the user to choose the objective function and the constraints depending on the situation at hand. Thus an optimizer that can accommodate general constraints must be chosen.

The good news is that there has been tremendous progress in general-purpose nonlinear optimization methods during the last two decades (see the book [32] for a good introduction to optimization methods; [33, 34, 35] for more advanced discussions and [36] for a practical guide on optimization software; also see the pair of articles [37, 38] for a discussion of unconstrained and constrained optimization methods, respectively; and finally [39, 40] for surveys of the state-of-the-art). Many pioneering attempts at circuit optimization during the late '60s and '70s are now being revisited since we have the optimization methods and computing horsepower to make those approaches practical for the first time.

The two best known large-scale optimizers that fit the bill are MINOS[41] and LANCELOT[12]. Further, by taking advantage of some features of the CUTE testing

environment[42], a user can easily benchmark many different nonlinear optimizers by expressing the optimization problem in the SIF language[12]. SIF allows the user to exploit *group partial separability*[12] in the statement of the tuning problem[22], which is particularly advantageous in static tuning. LANCELOT is best at solving highly nonlinear large-scale problems with nonlinear constraints. MINOS is advantageous when the constraints are linear or near-linear and the number of degrees of freedom at the solution is small.

5.2 Application of optimization methods

Nonlinear optimization packages are sophisticated software programs with a wealth of options. One can often reap rich awards by utilizing the available options wisely and by customization of the optimization code to the problem at hand. Hence, these packages must be applied not as black boxes but with care and if possible, with the help of a knowledgeable optimization person.

The circuit analysis carried out at each iteration of the tuning is more CPU intensive than the nonlinear optimization algorithm. Hence, every effort must be made during the optimization procedure to reduce the number of iterations, even at the cost of expensive computations. This situation leads to an interesting and unusual proposition for nonlinear optimization packages, which are usually tailored to analytic problems in which function and gradient evaluations are relatively inexpensive. Analytic problems do not exhibit noise in the data, whereas any simulation-based data is inherently noisy. Most nonlinear optimizers are not tailored to handle noise in the data and must be customized to some extent. The customization may consist of choices of settings or tolerances, or even modification of the algorithms. Great care must be taken in the choice of stopping criteria, tolerances for bound checks, tolerances on step sizes, initial choice of trust-region radius, etc.,[5].

Minimax optimization can be approached in a number of ways. The simplest is to remap the problem from

$$\begin{array}{lll} \text{minimize} & \text{maximum} & c_i(x) \\ x \in \Re^n & i \in \{1, 2, \cdots, m\} & \end{array} \tag{3}$$

to

$$\begin{array}{l} \text{minimize} \quad z \\ x \in \Re^n, z \\ \text{subject to} \quad z \geq c_i(x), \quad i \in \{1, 2, \cdots, m\}. \end{array} \tag{4}$$

The linear variable z is minimized, subject to the constraint that it is larger than each of the constituent $c_i(x)$ constraints; therefore, at optimality, z is as small as possible but larger than all the $c_i(x)$ values. Two "tricks" can be used to make the minimax optimization more effective. The first is to initialize z to the largest of the $c_i(x)$ values after the first function evaluation, thus starting off the problem in a feasible state. The second is to initialize the Lagrange multipliers corresponding to the constraints in (4) above to $1/m$, since the first-order Kuhn-Tucker conditions[34] dictate that the Lagrange multipliers must sum to unity at optimality.

In addition to providing gradients, if at all possible, Hessian (second partial derivative) information must be provided to the optimizer. In the absence of such information, the optimizer usually applies qausi-Newton low rank updates[35] on an initial guess (e.g., scaled unity matrix) of the Hessian, requiring a large number of iterations to build up good curvature information. Providing Hessian data to the optimizer dramatically reduces the number of iterations required to solve optimization problems.

The observation has been made that the optimizer usually introduces variables of its own to remap the problem when necessary. For example, slack variables are commonly introduced to convert inequalities to equality constraints. Further, new linear variables are added to accommodate minimax functions as in (4). These variables are independent of the problem variables, and hence any change in the sub-space of these variables does not require a re-evaluation of the circuit. One can thus consider *two-step updates*[43] or *spacer steps*[44] in this sub-space. Since these slack and pseudo-variables often occur in a known and simple functional form in the merit function of the optimizer, the merit function can be additionally optimized in the sub-space of these variables at each iteration of the optimization, without having to pay any additional circuit analysis cost. This "second step" has been found to improve the overall effectiveness of the optimization.

Two other types of optimization deserve mention. It has been argued that circuit tuning is a *multi-criterion optimization problem*[45] and what the designer really wants is the entire set of *Pareto optimal points*[46]. In practice, however, either repeated "goal-based" single-criterion optimization runs or repeated optimization runs with different weights on the various objective functions suffice to understand the inherent trade-offs involved. Second, the accommodation of *semi-infinite constraints* is a useful feature of any circuit tuner[1]. The statement of a semi-infinite problem is

$$\begin{array}{l} \text{minimize} \quad f(x) \\ x \in \Re^n \\ \text{subject to} \quad g(x, p) \leq 0 \quad \text{for all} \quad p^u \leq p \leq p^l. \end{array} \tag{5}$$

For example, the user may want to minimize area (f in (5) above), subject to the delay through the circuit (g) being less than a certain target, for *any* power supply value (the semi-infinite parameter p) in a certain range. Unfortunately, methods for multi-criteria optimization and accommodation of semi-infinite constraints (especially when multi-dimensional) make a tuner less efficient and much more complicated. These enhancements are not used in practice.

6 Special topics

6.1 Interconnect tuning

Performance optimization of interconnect is the subject of a separate paper by J. Cong et al in these proceedings, and so will be mentioned only briefly here. A survey of

interconnect optimization methods can be found in [47]. With finer metal lines, a higher fraction of the total delay is due to on-chip interconnect, and hence more attention needs to be paid to this part of the design. Simultaneous tuning of gates and interconnect[48, 49, 6] has been shown to yield superior results to tuning just the interconnect or just the drivers. Further, reduced order modeling of interconnect[50, 51, 52] has allowed the analysis of large interconnect networks in a computationally efficient manner. The availability of sensitivities[53] makes these macromodels attractive from a tuning perspective.

In practice, local nets and short global nets are usually sized to have the minimum allowed width. Long global nets are either made wider, or the delay improved by the manual insertion of buffers or repeaters. Interconnect tuners work best when they are tightly coupled to the floorplanner and layout data. Recently, many approaches to wire sizing, buffer placement and topology changes have appeared in the literature[47]. The challenge for the future is a holistic solution that combines all three approaches. Special considerations are in order for the design of power distribution[54] and clock distribution networks. In the case of clock distribution networks, the design is usually carried out in two phases[55]. First the topology of the clock network is determined. Then the buffers are placed and wires sized so as to both minimize the skew to the target points and reduce sensitivity to process variations. Transmission line effects may be important during this procedure[56].

6.2 Noise-aware tuning

Signal integrity and noise analysis are increasingly important and challenging aspects of high-speed digital design. Noise checking is usually carried out on a static basis in the same framework as static timing analysis[57]. However, noise-aware tuning can be applied to both static and dynamic methods. In the simplest case, additional noise constraints can be added to the tuner. These constraints can be derived from rules of thumb related to noise considerations, or from detailed analysis. The rules of thumb may take a simple form such as ratio-ing of device sizes. More formally, stability criteria[57] can be added to the tuning procedure as additional constraints.

There is usually a direct trade-off between noise immunity and performance. For example, the half-latch in a dynamic gate can be sized for aggressive performance, but there is a concomitant loss in noise margin. In digital circuits, channel lengths are usually not tuned; they are left at their minimum allowed size. However, in the context of the tradeoff between noise and performance, one could consider tuning device lengths. Tuning of device lengths can therefore play a significant role in noise-aware tuning (and, in fact, in statistical tuning since length variations are the primary cause of circuit-limited yield).

7 Avenues of future research

Tuning of circuits on a static-timing basis is one of the most challenging problems faced today. Maintaining sufficient timing accuracy while tuning large problems in the presence of arbitrary timing constraints is a difficult problem. Noise-aware tuning will gain prominence as more formal noise analysis methods are developed. Computation of Hessians of circuits will go a long way in making tuning more efficient. Further into the future, the solution of mixed continuous/integer problems will enable designers to optimize continuous variables simultaneously with discrete ones. Design centering and yield optimization require further work before they can enter the mainstream IC design process. For the brave, multi-criteria optimization and accommodation of semi-infinite constraints pose a long series of challenges.

8 Acknowledgments

The author owes a debt of gratitude to Andy Conn for generously sharing his deep knowledge of and long experience with nonlinear optimization. Ruud Haring was a partner in the author's circuit tuning efforts from the beginning and ensured that the tools developed were in line with designers' needs. Special appreciation is due to Sani Nassif for help with the "Statistical tuning" section, to Phillip Restle for his assistance on clock and interconnect tuning, and to Ken Shepard for help with the "Noise-aware tuning" section. The author would like to thank Prabhakar Kudva and Phillip Restle for suggestions on the manuscript. Finally, the author is grateful to the members of the JiffyTune and EinsTuner projects: Phil Strenski, Chai Wah Wu, Bill Wright, Nam Nguyen, Abe Elfadel, David Ling, Paula Coulman, Greg Morrill, Indraneel Das, Katya Scheinberg, Peter O'Brien, Jeff Soreff and Walt Molzen.

References

[1] W. Nye, D. C. Riley, A. Sangiovanni-Vincentelli, and A. L. Tits. DELIGHT.SPICE: An optimization-based system for the design of integrated circuits. *IEEE Transactions on Computer-Aided Design of ICs and Systems*, CAD-7(4):501–519, April 1988.

[2] R. K. Brayton and R. Spence. *Sensitivity and optimization*, volume 2 of *CAD of Electronic Circuits*. Elsevier Scientific Publishing Co., Amsterdam, The Netherlands, 1980.

[3] R. K. Brayton, G. D. Hachtel, and A. L. Sangiovanni-Vincentelli. A survey of optimization techniques for integrated-circuit design. *Proceedings of the IEEE*, 69(10):1334–1362, October 1981.

[4] J.-M. Shyu and A. Sangiovanni-Vincentelli. ECSTASY: a new environment for IC design optimization. *IEEE International Conference on Computer-Aided Design*, pages 484–487, November 1988.

[5] A. R. Conn, P. K. Coulman, R. A. Haring, G. L. Morrill, and C. Visweswariah. Optimization of custom MOS circuits by transistor sizing. *IEEE International Conference on Computer-Aided Design*, pages 174–180, November 1996.

[6] A. R. Conn, R. A. Haring, C. Visweswariah, and C. W. Wu. Circuit optimization via adjoint lagrangians. *IEEE International Conference on Computer-Aided Design*, November 1997.

[7] A. R. Conn, P. K. Coulman, R. A. Haring, G. L. Morrill, C. Visweswariah, and C. W. Wu. Jiffytune: circuit optimization using time-domain sensitivities. *IEEE Transactions on Computer-Aided Design of ICs and Systems*, 1997. Submitted for publication.

[8] D. A. Hocevar, P. Yang, T. N. Trick, and B. D. Epler. Transient sensitivity computation for MOSFET circuits. *IEEE Transactions on Computer-Aided Design of ICs and Systems*, CAD-4(4):609–620, October 1985.

[9] S. W. Director and R. A. Rohrer. The generalized adjoint network and network sensitivities. *IEEE Transactions on Circuit Theory*, CT-16(3):318–323, August 1969.

[10] L. T. Pillage, R. A. Rohrer, and C. Visweswariah. *Electronic circuit and system simulation methods.* McGraw-Hill, 1995.

[11] B. D. H. Tellegen. A general network theorem, with applications. *Philips Research Reports*, 7:259–269, 1952.

[12] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A).* Springer Verlag, 1992.

[13] C. Visweswariah and R. A. Rohrer. Piecewise approximate circuit simulation. *IEEE Transactions on Computer-Aided Design of ICs and Systems*, 10(7):861–870, July 1991.

[14] P. Feldmann, T. V. Nguyen, S. W. Director, and R. A. Rohrer. Sensitivity computation in piecewise approximate circuit simulation. *IEEE Transactions on Computer-Aided Design of ICs and Systems*, 10(2):171–183, February 1991.

[15] T. V. Nguyen, P. Feldmann, S. W. Director, and R. A. Rohrer. SPECS simulation validation with efficient transient sensitivity computation. *IEEE International Conference on Computer-Aided Design*, pages 252–255, November 1989.

[16] T. V. Nguyen. Transient sensitivity computation and applications. Technical Report CMUCAD-91-40, Carnegie Mellon University, Pittsburgh, PA, 1991.

[17] R. B. Hitchcock, Sr., G. L. Smith, and D. D. Cheng. Timing analysis of computer hardware. *IBM Journal of Research and Development*, pages 100–105, January 1982.

[18] J. P. Fishburn and A. E. Dunlop. TILOS: A posynomial programming approach to transistor sizing. *IEEE International Conference on Computer-Aided Design*, pages 326–328, November 1985.

[19] S. S. Sapatnekar, V. B. Rao, P. M. Vaidya, and S. M. Kang. An exact solution to the transistor sizing problem for cmos circuits using convex optimization. *IEEE Transactions on Computer-Aided Design of ICs and Systems*, CAD-12(11):1621–1634, November 1993.

[20] W. C. Elmore. The transient analysis of damped linear networks with particular regard to wideband amplifiers. *Journal of Applied Physics*, 19(1):55–63, 1948.

[21] P. Penfield and J. Rubinstein. Signal delay in rc tree networks. In *Proceedings of the 2nd Caltech VLSI Conference*, pages 269–283, March 1981.

[22] M. D. Matson and L. A. Glasser. Macromodeling and optimization of digital MOS VLSI circuits. *IEEE Transactions on Computer-Aided Design of ICs and Systems*, CAD-5(4):659–678, October 1986.

[23] R. J. Duffin, E. L. Peterson, and C. Zener. *Geometric programming - theory and applications.* John Wiley and Sons, New York, 1967.

[24] S. S. Sapatnekar and W. Chuang. Power vs. delay in gate sizing: conflicting objectives? *IEEE International Conference on Computer-Aided Design*, pages 463–466, November 1995.

[25] P. K. Sancheti and S. S. Sapatnekar. Optimal design of macrocells for low power and high speed. *IEEE Transactions on Computer-Aided Design of ICs and Systems*, CAD-15(9):1160–1166, September 1996.

[26] R. Spence and R. S. Soin. *Tolerance design of electronic circuits.* Addison-Wesley Publishing Company, 1988.

[27] J. C. Zhang and M. A. Styblinski. *Yield variablity optimization of integrated circuits.* Kluwer Academic Publishers, 1995.

[28] S. W. Director and W. Maly, editors. *Statistical approach to VLSI*, volume 8 of *Advances in CAD for VLSI.* North-Holland, 1994.

[29] C. Michael and M. Ismail. *Statistical modeling for computer-aided design of MOS VLSI circuits.* Kluwer Academic Publishers, 1993.

[30] G. Kjellstrom and L. Taxen. Stochastic optimization in system design. *IEEE Transactions on Circuits and Systems*, July 1981.

[31] J. G. Ecker. Geometric programming: methods, computations and applications. *SIAM review*, 22(3):338–362, July 1980.

[32] S. G. Nash and A. Sofer. *Linear and nonlinear programming.* McGraw-Hill, 1996.

[33] J. E. Dennis, Jr. and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations.* Society for Industrial and Applied Mathematics, Philadelphia, 1996.

[34] R. Fletcher. *Practical methods of optimization.* John Wiley and Sons, Chichester, second edition, 1987.

[35] P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization.* Academic Press, London and New York, 1981.

[36] J. J. Moré and S. J. Wright. *Optimization software guide.* Society for Industrial and Applied Mathematics, Philadelphia, 1993.

[37] J. E. Dennis, Jr. and R. B. Schnabel. Unconstrained optimization. In G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, editors, *Optimization*, volume 1 of *Handbooks in operations research and management science*, pages 73–170. North-Holland, Amsterdam, The Netherlands, 1989.

[38] P. E. Gill and W. Murray. Constrained optimization. In G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, editors, *Optimization*, volume 1 of *Handbooks in operations research and management science*, pages 73–170. North-Holland, Amsterdam, The Netherlands, 1989.

[39] A. R. Conn, Nick Gould, and Ph. L. Toint. Algorithms for large-scale constrained nonlinear optimization: a current survey. In E. Spedicato, editor, *Algorithms for continuous optimization: the state of the art*, pages 287–332, Dordrecht, The Netherlands, 1994. Kluwer Academic Publishers. Volume 434 of *NATO ASI Series C: Mathematical and Physical Sciences.*

[40] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Methods for nonlinear constraints in optimization calculations. In I. S. Duff and G. A. Watson, editors, *The state of the art in numerical analysis*, pages 363–390. Oxford University Press, 1997.

[41] B. A. Murtagh and M.A. Saunders. MINOS 5.1 user's guide. Technical Report SOL 83-20R, Stanford University, Stanford, CA, December 1983. revised January 1987.

[42] I. Bongartz, A. R. Conn, N. I. M. Gould, and Ph. L. Toint. CUTE: constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software*, 21(1):123–160, March 1995.

[43] A. R. Conn, L. N. Vicente, and C. Visweswariah. Two-step algorithms for nonlinear optimization with structured applications. Research Report (in preparation), IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, USA, 1997.

[44] D. P. Bertsekas. *Constrained Optimization and Lagrange Multipliers Methods.* Academic Press, London, 1982.

[45] M. R. Lightner and S. W. Director. Multiple criterion optimization for the design of electronic circuits. *IEEE Transactions on Circuits and Systems*, CAS-28(2):169–179, March 1981.

[46] P. L. Yu. Multiple criteria decision making: five basic concepts. In G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, editors, *Optimization*, volume 1 of *Handbooks in operations research and management science*, pages 663–699. North-Holland, Amsterdam, The Netherlands, 1989.

[47] J. Cong, L. He, C-K. Koh, and P. H. Madden. Performance optimization of VLSI interconnect layout. *Integration*, 21:1–94, January 1996.

[48] J. J. Cong and C.-K. Koh. Simultaneous driver and wire sizing for performance and power optimization. *IEEE Transactions on VLSI Systems*, 2(4):408–425, December 1994.

[49] N. Menezes, R. Baldick, and L. T. Pileggi. A sequential quadratic programming approach to concurrent gate and wire sizing. *IEEE International Conference on Computer-Aided Design*, pages 144–151, November 1995.

[50] L. T. Pillage and R. A. Rohrer. Asymptotic waveform evaluation for timing analysis. *IEEE Transactions on Computer-Aided Design of ICs and Systems*, 9:352–366, April 1990.

[51] P. Feldmann and R. W. Freund. Efficient linear circuit analysis by Padé approximation via the Lanczos process. *IEEE Transactions on Computer-Aided Design of ICs and Systems*, 14(5):639–649, May 1995.

[52] L. M. Silveira, M. Kamon, I. Elfadel, and J. White. A coordinate-transformed Arnoldi algorithm for generating guaranteed stable reduced-order models of RLC circuits. *IEEE International Conference on Computer-Aided Design*, pages 288–294, November 1996.

[53] R. W. Freund and P. Feldmann. Efficient small-signal circuit analysis and sensitivity computations with the PVL algorithm. *IEEE International Conference on Computer-Aided Design*, pages 404–411, November 1994.

[54] H. H. Chen and D. D. Ling. Power supply noise analysis methodology for deep-submicron VLSI chip design. *Proc. 1997 Design Automation Conference*, pages 638–643, June 1997.

[55] E. G. Friedman (Editor). *Clock distribution networks in VLSI circuits and systems.* IEEE Press, 1995.

[56] P. J. Restle, K. A. Jenkins, A. Deutsch, and P. W. Cook. Measurement and analysis of on-chip transmission line effects in a 400mhz microprocessor. *IEEE Journal of Solid State Circuits*, 1997. To be published.

[57] K. L. Shepard and V. Narayanan. Noise in deep submicron digital design. *IEEE International Conference on Computer-Aided Design*, pages 524–531, November 1996.