THE CENTER DESIGN OPTIMIZATION SYSTEM

Kishore Singhal, Colin C. McAndrew, Sani R. Nassif, and V. Visvanathan

CENTER is a software system based on the UNIX® operating system used to optimize integrated circuit designs and semiconductor device technologies. Its novel architecture and interface mechanisms allow rapid linking to both arbitrary simulators that define a particular design problem and to existing optimization software. This significantly reduces the time and effort required to solve individual design optimization problems. The architecture allows easy integration of sampling, approximation, and statistical design modules, and enables high-level tasks to be developed using the low-level modules as building blocks. In this article, we detail the functionality and architecture of CENTER and give examples of its use for design optimization.

Kishore Singhal, Colin C. McAndrew, and Sani R. Nassif *are in the Technology CAD Department of AT&T Bell Laboratories at Allentown-Cedar Crest, Allentown, Pennsylvania. Mr. Singhal is a supervisor and Messrs. McAndrew and Nassif are members of technical staff.* ***V. Visvanathan*** *is a distinguished member of technical staff in the Integrated Circuit Simulation Department of Bell Laboratories in Murray Hill, New Jersey. Kishore Singhal joined the company in 1985; his group is responsible for statistical characterization and optimization of integrated circuits, and compact model development and parameter extraction. Mr. Singhal received a B.Tech. from the Indian Institute of Technology, Kharagpur, India, and M.S. and Eng.ScD. degrees in electrical engineering from Columbia University. Colin McAndrew joined* (continued on page 92)

Introduction

Integrated circuit (IC) design and manufacture and semiconductor device technologies are important for the success of high-technology electronics. (See Panel 1 for a list of acronyms used in this paper.) Both IC design and semiconductor processing technology development are complex tasks; yet high-quality ICs must be designed and manufactured reliably and quickly. To optimize IC designs and technologies, and thereby produce high-quality ICs, we must determine the best values of many parameters controlling the design or manufacturing process. Examples of such parameters are: (a) the widths of transistors for an IC design and (b) implant doses and annealing temperatures for semiconductor manufacturing processes. The complexity of these design tasks can challenge even experienced IC designers and processing technologists. Therefore, to aid IC design optimization and the development of semiconductor processing technologies, we have developed the CENTER generalized design optimization system and used it with programs that simulate semiconductor manufacturing processes and semiconductor device and circuit behavior.

Panel 1. Acronyms in This Paper

ADVICE	aid in design verification for integrated circuit engineering.
BICEPS	Bell integrated circuit engineering process simulator.
CAD	computer-aided design.
CENTER	a generalized design optimization system.
CENTER/ ADVICE	an integrated circuit design optimization system.
CENTER/ BRIDGE	tools for data translation and integration in the CENTER system.
CMOS	complementary metal-oxide semiconductor.
CPU	central processing unit.
FABRICS	an integrated circuit fabrication process simulator.
IC	integrated circuit.
MECCA	modeling, extraction and confirmation for circuit analysis.
MEDUSA	a semiconductor device simulator.
MINOS	a system for solving large-scale optimization problems.
MOSFET	metal-oxide semiconductor field-effect transistor.
SAO	sample-approximate-optimize.
SPICE	simulation program with integrated circuit emphasis.
WATOPT	an optimizer for circuit applications.

We have applied the CENTER system to such design problems as:

- Optimization of transistor widths for standard cells used in IC design to maximize speed and/or minimize area.
- Optimization of capacitances, resistances, and transistor sizes for high-performance analog circuits to maximize gain and bandwidth and minimize ripple.
- Optimization of register lengths for digital phase-locked loop circuits to minimize register lengths and output jitter.
- Optimization of silicon IC manufacturing processes to maximize transistor gain and current drive capability.
- Optimization of gallium arsenide IC manufacturing processes and transistor widths to maximize switching speeds and minimize sensitivity to electrical noise.
- Extraction of parameters for analytic models for circuit simulators to minimize differences between model predictions and measurements.

The literature is rich on the applicability of optimization techniques to circuit design. (See Brayton et al.[1] and Nye et al.[2] for a historical review.) However, these techniques have been used only to a limited degree by the circuit design community. This is because the available software tools suffer from one of two drawbacks: either a limited range of applicability or a complex software architecture that makes interfacing different simulators and/or numerical optimization packages difficult.

In designing the CENTER system, we wanted to simplify the effort required in applying optimization software to individual design optimization problems. Although many good software packages for numerical optimization exist, linking them to specific design optimization problems can be difficult and time-consuming. CENTER has a unique architecture and a suite of programs for data translation and integration (the CENTER/BRIDGE tools) that allow simple, rapid connection of optimization software to specific design tasks. The architecture uses well-defined interfaces and takes advantage of features of the UNIX system.

CENTER also addresses another aspect of practical design optimization: namely that, instead of the generation of rigorously optimal designs, design improvement or near-optimal design is usually sufficient. Typically, design performance is evaluated using a computer program that simulates the behavior of the system under design. The simulation is a predictive model of the system and is generally expensive to compute. Design optimization only makes sense to an accuracy comparable to that of the simulator, which is commonly several percent. This is "loose" in terms of strict mathematical optimization. Therefore, although CENTER contains software for numerical optimization and is termed a design optimiza-

tion system, it is often used to generate *near*-optimal designs. CENTER can efficiently sample and build internal approximations to design performance measures as functions of design control parameters and apply optimization software to the approximations as well as directly to simulation programs. This leads to good design improvement at a fraction of the computational cost of conventional optimal design.

In this article, we detail the architecture and function of CENTER and give examples of its use for IC design and semiconductor processing technology optimization. The architecture provides an effective mechanism for linking statistical design software to practical design problems; thus, we also describe some other capabilities of CENTER: worst-case performance analysis and large-scale sensitivity analysis.

Our terminology is as follows: A system under design is defined by n control parameters, called *design variables* and denoted by the vector $\mathbf{x}$, and m performance measures, called *design performances* and denoted by the vector function $\mathbf{z}(\mathbf{x})$. We use the term design performances and the symbol $\mathbf{z}$ to include functions, called *design objectives*, to be minimized or maximized, as well as functions, called *design constraints*, that must satisfy specified constraints. A vector function of *approximations* to the design performances is denoted by $\hat{\mathbf{z}}(\mathbf{x})$. All design variables are *bounded* by upper and lower bounds. The n-dimensional hypercube formed by these bounds on design variables defines the *design space*. Design constraints can be specified as equality constraints, upper or lower bound constraints, or range constraints. For optimization, the *design goal* is to optimize the design objectives, subject to the specified design constraints. The entity that evaluates the design performances $\mathbf{z}(\mathbf{x})$ is the *simulator*. Typically, existing simulators have a data format and command syntax that do not conform to the CENTER $\mathbf{z}(\mathbf{x})$ evaluation protocol. Data *filters*, such as the CENTER/BRIDGE tools and programs in the `awk` language,[3] are used to transform information being passed from CENTER to the simulator and from the simulator back to CENTER to ensure that it is correctly formatted. The combination of data filters and simulator that conforms to the CENTER protocol is termed a *virtual simulator*.

The Functionality of CENTER

CENTER contains software modules for a range of functions; however, optimization is its major application.

Optimization. There are six numerical methods for nonlinear optimization presently available:

- Sequential quadratic programming
- A projected, augmented Lagrangian algorithm
- A quasi-Newton solver
- A nonlinear least squares minimizer
- The Nelder-Mead simplex method
- Simulated annealing.

Six methods are included because each has different areas of strength. This allows selection of an optimization strategy that is best suited to each design problem. The optimizers have been applied to problems with up to fifty design variables and twenty design performances.

Sequential quadratic programming is a modern, powerful optimization method and CENTER incorporates the WATOPT optimization code.[4] WATOPT is a fast, nonlinear, constrained, multiobjective optimizer specifically targeted for circuit design. WATOPT seeks to minimize a vector function, subject to equality and inequality constraints, by generating the associated Lagrange function and iteratively determining its stationary point by taking steps along search directions corresponding to the minima of a constrained quadratic function. The variable metric matrix of the quadratic is revised using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula.[5] WATOPT accounts for multiple design objectives by seeking a min-max point of the Pareto optimum.[6] It seeks the vector $\mathbf{x}$, which minimizes the maximum relative deviation of the design objectives from their separately attainable minima. Further details of WATOPT are available in Reference 4.

For large-scale optimization problems, CENTER

incorporates the MINOS system.[7] MINOS can solve linear programming problems, linearly constrained nonlinear programming problems, and nonlinearly constrained nonlinear programming problems. IC design and semiconductor device technology optimization problems fall into the last category, for which MINOS uses a projected, augmented Lagrangian method. MINOS minimizes a scalar function of the design variables subject to equality and inequality constraints. CENTER generates the scalar function as a weighted sum of the design objectives. MINOS is a widely used, robust optimization code particularly suited to constrained optimization problems with many design variables.

The quasi-Newton solver, like WATOPT, uses the BFGS update for the Hessian, but unlike WATOPT, it uses a trust-region technique to promote convergence from poor starting points. The technique limits steps taken during the iterative solution to a region of the design space in which the quadratic model, associated with Newton-like methods, is a reliable approximation that can be "trusted." Details of the quasi-Newton method are given in Reference 8. The solver seeks a minimum of a scalar function of the design variables and it does not directly handle constraints. Instead, the scalar cost function computed by CENTER consists of a weighted sum of the design objectives to which constraint violations are added as penalty terms. The quasi-Newton solver is robust and has proved to be an effective optimization code.

The nonlinear least squares minimizer is also a variant of Newton's method that uses a trust region.[9] The model Hessian is chosen adaptively; part is computed exactly and part is approximated by a quasi-Newton updating method. The minimizer considers **z** as a vector function of residuals whose sum of squares is to be minimized. It does not handle constraints. The algorithm can reduce to a Gauss-Newton or Levenberg-Marquardt method (e.g., see Reference 5, pp. 523-528), but, on problems with large residuals, it often works better than these methods. The minimizer is, therefore, used for data fitting rather than general, constrained optimization.

The Nelder-Mead nonlinear simplex optimization method is a simple algorithm that, in certain circumstances, works well. A simplex is a bounded polytope with $n + 1$ vertices in the n-dimensional design space. The method, described by Press et al.,[5] consists of moving vertices of the simplex by reflection, expansion, and contraction so that the simplex contracts around the optimal point. The rate of convergence of the method is slow, because it uses no gradient information. However, it can often generate a near-optimal point rapidly. Multiple objectives and penalties for constraint violations are merged into a composite scalar cost function for the Nelder-Mead optimizer.

Simulated annealing involves random sampling in the design space and keeping track of the point that generates the best design performances. As the algorithm progresses, the extent of the space over which the random sampling is done is continually shrunk about the current best point. The shrinking is analogous to decreasing temperature in a physical system—hence the name simulated annealing. We have modified the simulated annealing algorithm originally intended for combinatorial problems[10] to work on mixed continuous-discrete variable problems. Simulated annealing is conceptually simple and, although it can only find near-optimal solutions, it works well in practice. An important feature of simulated annealing is that uphill moves are sometimes allowed. Thus, a point with a worse performance than the current best point can be accepted. This allows the algorithm to jump out of local optima and have a better chance at finding the global solution to a design problem, rather than being "greedy" and trying to move towards a local optimum as fast as possible (as with gradient-based optimization methods). Multiple objectives and penalties for constraint violations are merged into a composite scalar cost function for the simulated annealing optimizer.

The composite scalar cost function used by the quasi-Newton, Nelder-Mead nonlinear simplex and

Table I. Comparison of CENTER Optimizers

Method	WATOPT	MINOS	Quasi-Newton Method	Least Squares Minimizer	Nelder-Mead Simplex	Simulated Annealing
Gradients required?	Yes	Yes	Yes	Yes	No	No
Fast initial improvement?	No	No	No	No	Yes	Yes
Fast final convergence?	Yes	Yes	Yes	Yes	No	No
Type of optimum	Local	Local	Local	Local	Local	Global
Multiobjective criterion	Pareto	Composite	Composite	Least Squares	Composite	Composite
Constraint handling	Lagrangian	Lagrangian	Penalty	None	Penalty	Penalty

simulated annealing optimizers is computed as follows: Design objectives to be minimized or maximized are added to and subtracted from the cost function respectively, and penalty terms are added for violations of constraints. The composite cost uses weights that reflect the relative importance of each design objective and constraint and that incorporate user-supplied estimates of the range of the objectives and constraints for scaling. This allows meaningful comparison of such quantities as doping concentrations, threshold voltages, and rise times, which are of order 10^{18}, 10^{0}, and 10^{-12}, respectively. Because of the use of penalty terms, the optimizers that use the composite cost function cannot guarantee that the constraints are strictly satisfied. This is rarely a problem in practice because constraints are usually "soft." However, all optimizers are coded so that the bounds specified for design variables are not violated.

We use a number of optimization methods because no single optimization method works best on all design problems. Table I summarizes various features of the different methods.

If an optimizer requires gradients and they are not available, CENTER will calculate them by differencing. The strengths and weaknesses of the optimizers lead to the following observations:

- WATOPT or MINOS should be used where constraints must be satisfied.
- Simulated annealing or the Nelder-Mead method should be used when the design performances do not have well-defined derivatives with respect to the design variables. This occurs when $\mathbf{z}(\mathbf{x})$ is discontinuous or noisy.
- Simulated annealing should be used for problems that exhibit multiple optima.
- The nonlinear least squares method should be used for data fitting, such as when extracting parameters of analytic models for circuit simulators.
- MINOS should be used for large-scale problems.
- Simulated annealing should be used for problems that involve discrete design variables.

Another advantage of integrating several optimizers in a single system is that hybrid optimization strategies can be built through sequential application of individual optimizers. For example, simulated annealing can escape from local optima, which is desirable. But, once it generates a point near a global optimum, it approaches the optimum slowly. In contrast, gradient-based methods are trapped by local minima, but converge rapidly when near a minimum. Thus, application of simulated annealing followed by WATOPT exploits the strengths of both methods and does better than either method alone.

Sampling and Approximation. One feature of CENTER is its ability to sample a design space and build mathematical approximations, $\hat{\mathbf{z}}(\mathbf{x})$, of design performances as functions of design variables. The optimization algorithms can work with approximated performances, $\hat{\mathbf{z}}(\mathbf{x})$, as well as with simulated performances, $\mathbf{z}(\mathbf{x})$. We call this the SAO (sample-approximate-optimize)

strategy. Although the strategy may not find a true optimum of $\mathbf{z}(\mathbf{x})$, it achieves the practical end of design improvement and requires substantially less computing effort than conventional optimization. This is because the cost of an optimization is typically dominated by $\mathbf{z}(\mathbf{x})$ simulation; by comparison, the cost of execution of the optimization code and the overhead of approximation building and evaluation are small. The sampling and approximation building strategy reduces the number of $\mathbf{z}(\mathbf{x})$ evaluations required for an "optimization."

CENTER contains several methods for sampling a design space, including Hadamard sampling[11] and Latin Hypercube sampling.[12] Both methods generate samples that are well spread over the design space. A Hadamard matrix has elements with value either +1 or −1 and rows that are mutually orthogonal. An example of a Hadamard matrix, of order 8, is shown below.

$$\begin{bmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 \\ +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 \\ +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 \end{bmatrix}$$

For sampling a design space, the rows of the matrix are treated as vectors of design variables, with −1 and +1 signifying that the associated design variable assumes its lower or upper bound value respectively. The first column of the matrix is not used for sampling. Hadamard sampling is parsimonious because it generates only slightly more than n samples for an n-dimensional design space and is especially useful when the design performances are monotonic functions of the design variables. Latin Hypercube sampling requires roughly $3n$ samples; thus, it is still efficient and complementary to Hadamard sampling in that it generates samples within the interior of the hypercube bounding the design space. Coverage of both the boundary and interior of the design space can thus be achieved by a combination of Hadamard and Latin Hypercube sampling. The Latin Hypercube method partitions the design space into bins and generates samples within selected bins using random sampling. This generates statistical samples that span the design space more efficiently than straightforward Monte Carlo sampling.

Several types of $\hat{\mathbf{z}}(\mathbf{x})$ approximations can be built, including interpolating Gaussians,[13] multinomial regression models, and spline models. Given $\mathbf{z}(\mathbf{x}^{(k)})$ for points $k = 1, \cdots, K$, the interpolating Gaussians approximate a particular design performance z_j at a point $\mathbf{x} = [x_1\ x_2\ \cdots\ x_n]^T$ by

$$\hat{z}_j = \mu_j + \sum_{k=1}^{K} \gamma_k \exp\left[-\sum_{i=1}^{n} \frac{(x_i^{(k)} - x_i)^2}{2\rho^2}\right]$$

where μ_j is the average value of z_j over the samples, ρ controls the widths of the Gaussians, and the coefficients γ_k are computed so that $\hat{z}_j$ matches the values at the known points. Multiple minima and maxima can be modeled by Gaussian approximation because it is an exact method for interpolating multivariate functions.

Besides enabling near-optimal designs to be generated efficiently, the use of approximated rather than simulated values of design performances has another important implication. Simulation of design performances is usually noisy, which impairs the convergence of gradient-based optimizers when gradients are computed by numerical differencing. The approximations are chosen to be continuous and differentiable and, thus, work well with gradient-based optimizers. The architecture of CENTER, as discussed in the following section, allows rapid integration of new sampling methods and approximation techniques.

Although the SAO strategy yields design improvement, the procedure can be refined. (See

Figure 1.) After finding an optimum, $\mathbf{x}^*$, predicted by an approximation, $\hat{\mathbf{z}}(\mathbf{x}^*)$, the simulated design performance, $\mathbf{z}(\mathbf{x}^*)$, is evaluated and compared against the approximator prediction. If the prediction is poor, the newly evaluated simulation is included in the data set to build a new approximation and the optimum of the updated approximation is determined. The procedure is repeated until the approximated and simulated design performances are sufficiently close, at which point it is stopped. This iterative application of the SAO technique yields near-optimal solutions with substantially reduced effort, compared to applying the optimization codes directly to the $\mathbf{z}(\mathbf{x})$ simulator.

Other Functional Capabilities. CENTER can also do other analyses useful for the evaluation of IC designs and semiconductor fabrication processes. Worst-case performance analysis can be done by using Hadamard sampling and then using the approximation capability to build a linear regression model of $\mathbf{z}(\mathbf{x})$.[14] The regression model can be used to predict the best-case and worst-case corners of the hypercube that bounds the n-dimensional design space. The method relies on the orthogonality of the Hadamard samples and the relevant corners follow trivially from the signs of the coefficients of the linear model.

Large-scale sensitivity analysis is done by generating large variations around a specified operating point in the design space and analyzing the resulting variations in $\mathbf{z}$. The design variables are then ranked according to their relative influence on the design performances. Large-scale sensitivity analysis differs from small-scale sensitivity analysis in the size of the perturbations of the design variables, and it provides information about the sensitivity of a design to manufacturing variations. For example, it allows the identification of designs that may be optimal but lie near a region of failure, so that fluctuations inherent in manufacturing would result in poor yield.

Other features of CENTER include control via menus, the ability to run in either interactive or batch modes, and the capability to record and replay sequences of actions in journal scripts. The control menus are full-screen forms that support an interactive help facility and have powerful error checking capabilities. They are distinct from the functional task modules of CENTER. This allows the user interface to be easily upgraded as more advanced interface mechanisms evolve.

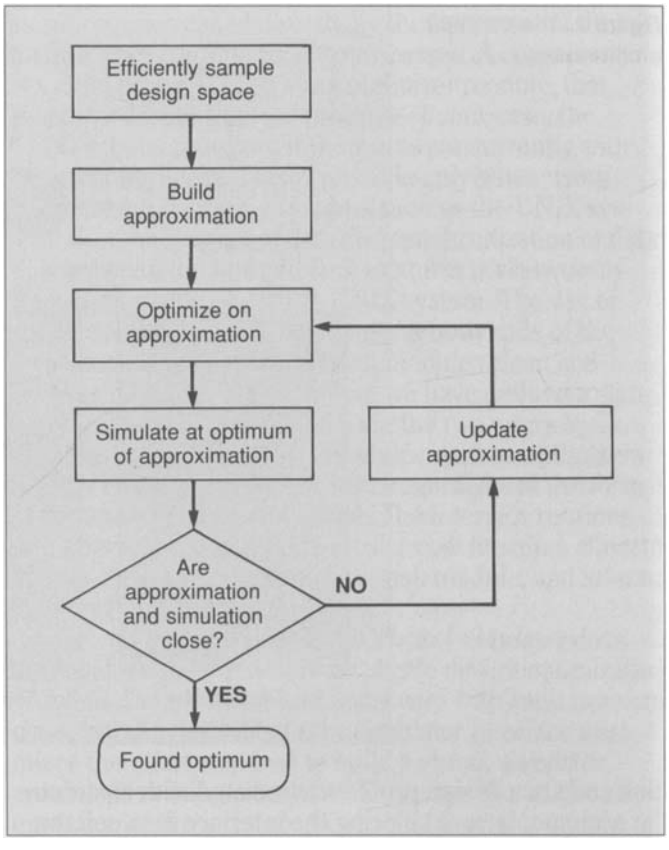
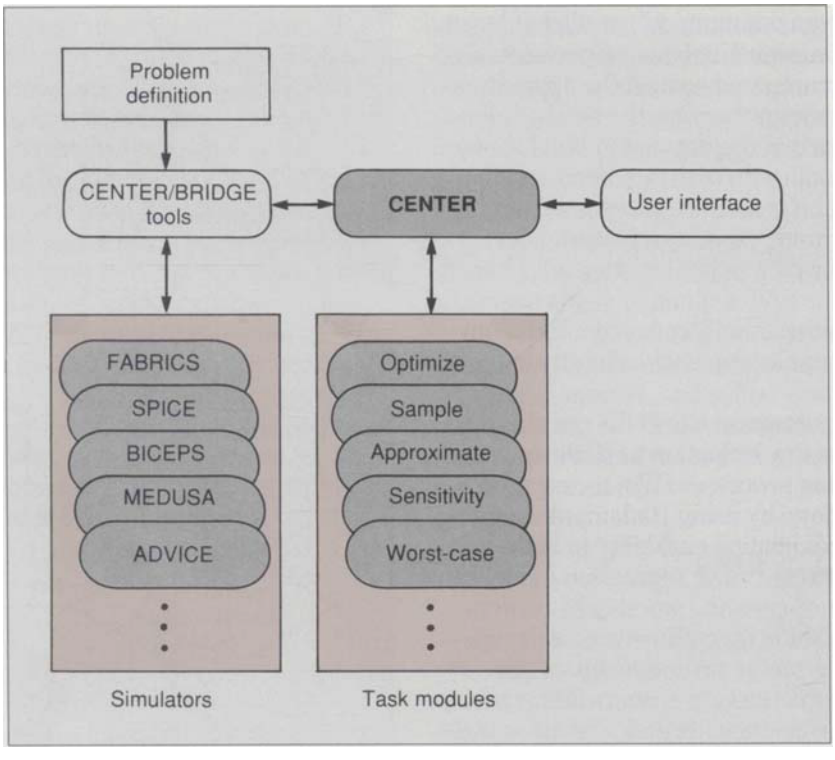


Figure 1. The iterative SAO strategy.

The CENTER Architecture

As mentioned above, the most difficult and time-consuming task when trying to apply a specific optimiza-

Figure 2. The CENTER architecture.



tion code to a design problem associated with a particular $\mathbf{z}(\mathbf{x})$ simulator is tailoring the interface between the simulator and optimizer. The architecture and functionality of CENTER and the CENTER/BRIDGE tools are designed to minimize the effort required for such an interface. Figure 2 shows the general structure of the architecture. The control portion, labeled CENTER, acts as a token-ring software bus that routes $\mathbf{x}$ and $\mathbf{z}$ or $\hat{\mathbf{z}}$ data between samplers, approximation builders and evaluators, optimizers, the simulator interface, and the other task modules. The CENTER/BRIDGE tools are, in part, filters for data translation that allow design variables, $\mathbf{x}$, generated by CENTER, to be transformed into a format compatible with a simulator and enable simulator outputs to be analyzed to produce design performances, $\mathbf{z}$, in the format required by CENTER. The major interfaces are from the bus to a task module, such as an optimizer, and from the bus to the $\mathbf{z}(\mathbf{x})$ simulator.

The Task Module Interface. In Figure 2, when task modules request a $\mathbf{z}(\mathbf{x})$ or $\hat{\mathbf{z}}(\mathbf{x})$ evaluation, they must do so via the bus. The modules, therefore, use a reverse-call mechanism. A module executes a return statement with a flag set to signal the bus that it requires a function evaluation for the $\mathbf{x}$ it has provided. The bus requests the necessary $\mathbf{z}(\mathbf{x})$ simulation or $\hat{\mathbf{z}}(\mathbf{x})$ approximation and passes the evaluated design performances back to the module, which continues its task. Other functions of the bus are: scaling of design variables, which greatly improves the performance of the optimizers; interactive alteration of the optimization problem definition, such as

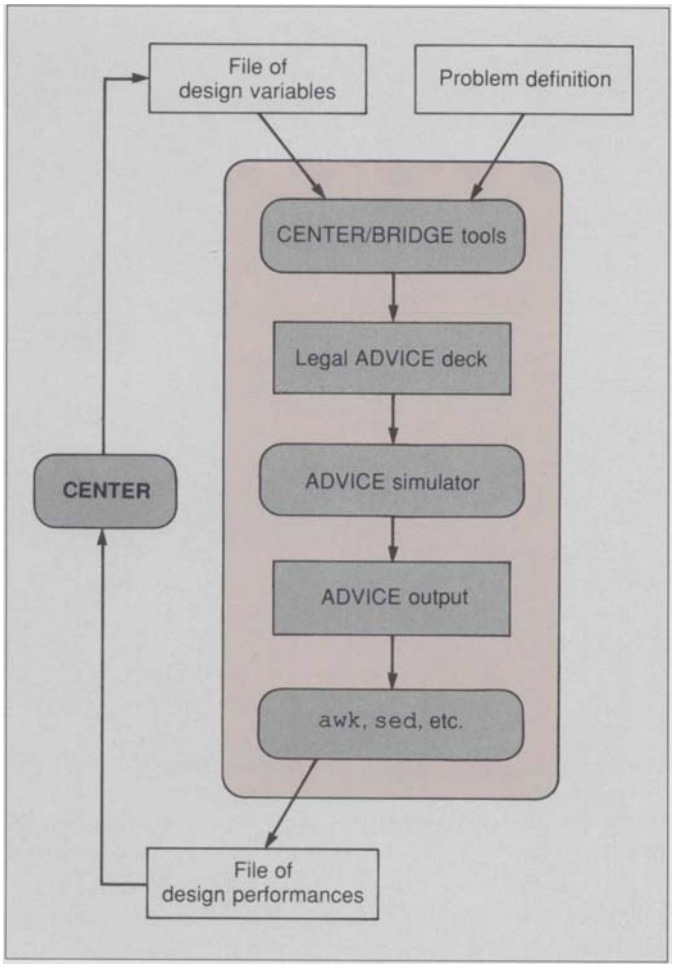


Figure 3. The ADVICE virtual simulator in `shell` mode.

excluding specific design variables from an optimization; and mapping between a subset of optimization variables and the complete set of design variables. These functions are done by data translation modules that are called by the bus prior to a call to a particular task module.

Although some task modules are implemented as subroutines called directly by the bus, most of the modules are configured as coprocesses. A coprocess is a child process, such as an optimizer module, that is spawned from a parent process—in our case, the CENTER bus program. It then runs concurrently with the parent process. This is possible only when using multitasking operating systems such as the UNIX system. Communication of data and synchronization of data flow between the bus and task modules is via two-way `pipes`, available under the UNIX system. The use of small, standard interface routines at both ends of the `pipe` makes the link to the task modules clean and ensures modularity. In addition, we have defined a standard protocol for passing data via the two-way `pipes`. This has made adding new modules such as optimizers to CENTER a simple procedure, regardless of the form of the underlying module code. The interface routines also generate the composite scalar cost function, allocate dynamic memory required by a task module, and interact with users via full-screen forms.

The Simulator Interface. A *virtual simulator* does $\mathbf{z}(\mathbf{x})$ evaluations relevant to a specific design optimization problem. Details of the simulator vary with each problem class, but the flexibility of the simulator interface minimizes the effort required to build a virtual simulator around an existing simulator. There are two modes of operation of the interface (a `shell` mode and a `pipe` mode) and three types of actions: initialization, $\mathbf{z}(\mathbf{x})$ simulation, and termination. The design problem specification is provided during initialization; the termination state is self-explanatory. The heart of the interface caters to $\mathbf{z}(\mathbf{x})$ simulation. To show the `shell` mode of operation of the interface for $\mathbf{z}(\mathbf{x})$ simulation, consider a concrete example relevant to IC design in which the performance of a circuit is being evaluated using the ADVICE circuit simulator.[15] Figure 3 outlines the procedure.

First, the vector of design variables is written to a file. CENTER then issues a command to execute a procedure (here, a UNIX system `shell` procedure) that maps the file containing the design variables into a file

containing the design performances. When the procedure ends, the file of design performances is read. The operation of the `shell` procedure is as follows: First, the file of design variables and a problem definition file are used to generate a valid ADVICE circuit file. The `shell` procedure uses the CENTER/BRIDGE tools to do this task. The data formats and/or command syntaxes of typical simulators such as ADVICE are generally too restrictive to allow inclusion of information relevant to design optimization. We have defined a general language extension that enables design variable specifications to be added to the command and/or data files of arbitrary simulators to form optimization problem definition files. The CENTER/BRIDGE tools replace the constructs defining design variables, which are embedded in the problem definition files, with the appropriate numerical values from the file of design variables generated by CENTER. This yields valid simulator command and/or data files. (The CENTER/BRIDGE tools also analyze the problem definition file and generate the design problem specification for the initialization phase.)

Second, the `shell` procedure invokes ADVICE and instructs ADVICE to read the valid circuit file and do the required simulations.

Finally, the `shell` procedure uses standard UNIX system utilities such as `awk` and `sed` and, perhaps, the CENTER/BRIDGE tools, to extract the required circuit performance measures from the ADVICE output and place them in the design performance file. The `shell` procedure is a virtual simulator that, as part of its function, uses the specific simulator relevant to a particular design problem (here, ADVICE).

For the `pipe` mode interface, CENTER communicates with the **z(x)** simulator configured as a coprocess. Data communication is via a two-way `pipe` rather than files. Standard interface routines that conform to the `pipe` data passing protocol are available. Thus, if a subroutine, rather than a stand-alone simulator, is provided for **z(x)** evaluation, it is compiled and linked with a `pipe` interface routine to generate a coprocess. This, in effect, mimics dynamic linking and is done transparently by CENTER.

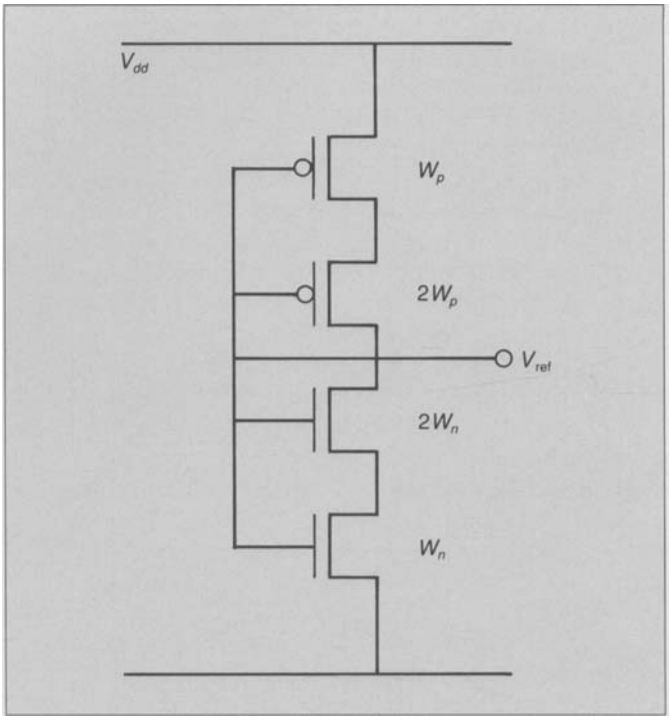


Figure 4. The simple CMOS voltage reference.

Intermediate Results Display. The other interesting feature of the CENTER architecture is the way the intermediate results of an optimization are presented to a user. A coprocess exists for the intermediate results display. When updated information is available, a UNIX system interprocess signal is sent to interrupt the display coprocess and new data is passed to the coprocess via a two-way `pipe`. The coprocess updates the displayed information and allows user interaction, such as scrolling, with the intermediate results. CENTER tasks can continue execution while intermediate results are viewed.

The CENTER architecture is, thus, flexible yet

simple. The well-defined task module and virtual simulator interfaces make integration of new optimization software and coupling to new $\mathbf{z}(\mathbf{x})$ simulators straightforward tasks. As noted previously, this was a major goal in developing CENTER.

CENTER for Circuit Design Optimization

CENTER has been applied to many types of design problems, but a major application is the optimization of circuit designs.

A Voltage Reference Design. Consider optimization of the simple complementary metal-oxide semiconductor (CMOS) voltage reference circuit in Figure 4. For IC design, the manufacturing process is fixed (in this example, as a CMOS technology). A designer generates a specific topological connection of transistors to do a required function. The parameters available to tune a design are the geometries of the transistors, their widths and lengths, although often the lengths are chosen from a small set of available sizes and are fixed thereafter. The design variables of the voltage reference circuit are, therefore, transistor widths. These are denoted as W_p and W_n; the widths of the *p*- and *n*-channel metal-oxide semiconductor field-effect transistors (MOSFETs) in Figure 4 are W_p, $2W_p$, $2W_n$, and W_n from top to bottom, respectively. V_{dd} denotes the supply voltage, nominally 5.0 volts, for the circuit in Figure 4. The design goal is to produce an output reference voltage, V_{ref}, of 2.5 volts.

Although the circuit is simple, the output voltage depends on the MOSFET characteristics and the load that the circuit drives. Selecting W_p and W_n to meet the design goal requires simulating the performance of the circuit as follows: The FABRICS simulator[16] is invoked to take the information about transistor widths and other information supplied about the processing operations used during fabrication of transistors and generate parameters for MOSFET models for the SPICE circuit simulator.[17] SPICE is then used to evaluate V_{ref} of the circuit for a specified supply voltage, ambient temperature, and load current.

Application of CENTER to the design, specifying a single equality constraint on the output of the voltage reference circuit, yielded a design with a V_{ref} of 2.500 volts in 0.25 CPU hours on a Sun Microsystems 3/260 workstation. This is a simple example of nominal case design that was solved easily. In practice, however, nominal design is only a part of the overall design procedure; it is imperative to consider how designs produced by real manufacturing processes fare in real operating environments. Therefore, the challenge in designing the circuit is to tackle a robust or statistical design—e.g., design a circuit with a V_{ref} as close as possible to 2.5 volts, taking into account manufacturing fluctuations and operating condition variations that the circuit will actually experience.[18]

To do a robust design, we identified the 10 most critical control parameters of the manufacturing process (including implantation doses and annealing temperatures and times) and generated a statistical sample of 50 combinations of these parameters. The sample was generated using the Latin Hypercube method; the statistical distributions of the parameters were accounted for during sampling. We also accounted for variations in the environment in which the circuit is required to operate by specifying three values each for the circuit supply voltage, load current, and the ambient temperature. This necessitated 450 FABRICS/SPICE simulations to evaluate the design for a set of specified transistor widths, rather than requiring a single simulation.

The goal of the robust design was to minimize the mean squared deviation of the 450 simulated output voltages from 2.5 volts. Because of the vast increase in the computational burden associated with the transition to robust design, we elected to apply the SAO strategy to the problem. For comparison, the nominal design had an average V_{ref} of 2.413 volts over the 450 samples and a yield of 29 percent, where yield was defined as the fraction of circuits whose output voltage was between 2.4 and 2.6 volts. In 2.75 CPU hours, the SAO strategy yielded a circuit with an average output voltage of 2.526 volts and

Table II. Results of Voltage Reference Circuit Optimization

Case	W_p	W_n	V_{ref} for nominal conditions	Average V_{ref} of 450 samples	Yield (percent)	CPU (hours)
Initial	5.00	3.00	2.463	2.374	27	
Nominal	5.34	2.54	2.500	2.413	29	0.25
SAO strategy	7.08	2.40	2.615	2.526	35	2.75
Simulated annealing	2.72	1.10	2.573	2.515	41	25.0

a yield of 35 percent. Interpolating Gaussians were used, and because such approximations account for multiple optima, the optimization strategy chosen was that of simulated annealing, followed by the quasi-Newton solver.

In contrast, when the simulated annealing algorithm was applied directly to the problem, we found a design with an average output voltage of 2.515 volts and a yield of 41 percent. However, the optimization took 25 CPU hours. Note that the SAO strategy did not find the same solution as the simulated annealing optimization because the initial samples used in building the approximation did not pick up a small valley in the $\mathbf{z}(\mathbf{x})$ surface. The SAO strategy did, however, provide good design improvement with a reasonable expenditure of computational effort.

Table II is the results of the various design strategies. The fourth column of the table lists the output voltage V_{ref} for the circuit manufactured with the process parameters and operating environment conditions at their nominal values. The fifth column lists the average of the values for the 450 samples of processing and operating conditions. The initial case represents the design before application of CENTER; the nominal case represents the results of CENTER for the design for nominal processing and operating conditions. The last two cases show the results of robust design using the SAO strategy and simulated annealing. Interestingly, the nominal output voltage for the SAO robust design is outside the acceptable range of 2.4 to 2.6 volts. Nevertheless, the SAO design is statistically superior to the nominal design because it has an average output voltage closer to 2.5 volts and a greater yield. More circuits manufactured with the SAO design will have output voltages within the 2.4 to 2.6 range than those manufactured with the nominal case design. The distributions of the output voltages for the 450 samples are shown in Figure 5 for both the nominal design and the simulated annealing design.

Our seemingly simple voltage reference design problem highlights several valuable points:

- First, it shows the importance of robust design. In practice, it is the performance of products with manufacturing variations and operating in different environments that is of importance, rather than the performance of a nominal product in a standard environment.
- Second, it shows that robust design can be substantially more complex and costly, in terms of computation, than nominal design. The SAO strategy is targeted towards efficient, near-optimization of robust design problems.
- Third, a fundamental limitation of the sample and approximate method has been highlighted: the low-density sampling done to make the method efficient may miss some valleys of the performance response surface. The simulated annealing algorithm works well for such cases, at considerably greater computational cost. (Gradient-based optimizers started in the "wrong" spot also miss these valleys and, thus, find a local, rather than a global, optimum.)
- As a final point, note that accounting for manufacturing process variations in the design problem specifica-

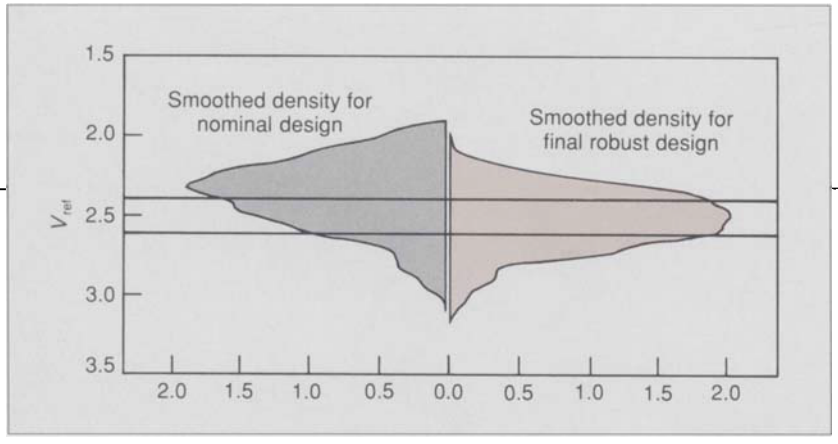


Figure 5. V_{ref} smoothed density estimates for nominal design and final robust design.

tion allows the sensitivity of the final design to the manufacturing process variations to be determined. This provides information about the most efficient way to deploy resources to improve the yield of the manufacturing process.

In this example, we have presented a simple implementation of robust design. More complex algorithms exist. (See, for example, Reference 19.)

A Repeater Circuit Design. A version of CENTER has been linked with the ADVICE circuit simulator and the resulting CENTER/ADVICE system has been used for circuit optimization. As an example, we use the optimization of a repeater circuit for an undersea lightwave system. For the repeater, the design goal is to maximize bandwidth without appreciable loss in gain, while keeping ripple to a minimum. [The repeater is an amplifier with a nearly constant gain for frequencies within a certain range (called the passband) and a rapid drop in gain with frequency for frequencies extending above and below the passband. Ripple is a measure of variation in gain for frequencies within the passband. It is desirable to have circuits with high gain, wide bandwidth of the passband, and little ripple.] The design variables are three resistor values, three capacitor values, and two transistor sizes. The initial design had a bandwidth of 347 megahertz (MHz), a gain of 75.4 decibels (dB), and exhibited virtually no ripple.

The design goal was loosely specified initially; hence the design proceeded in stages so that the performance of the evolving design could be evaluated by the designers. Table III shows the results of the optimization.

Initially, the simulated annealing method was applied, with a design goal of maximizing bandwidth and constraining gain to greater than 70 dB. This led to a design with a bandwidth of 670 MHz and a gain of 72.9 dB, but with 6.6 dB of ripple. By additionally constraining the gain at 550 MHz to be less than the mid-frequency gain, WATOPT produced a design in which the ripple was reduced to 2.9 dB with only a slight drop in bandwidth to 644 MHz and with virtually no loss of gain. Adding further constraints to reduce ripple allowed WATOPT to yield the final design shown in Table III. The approximate CPU times quoted in the table are for a VAX™ 11/785 computer. (VAX is a trademark of Digital Equipment Corporation.) The improvement of the design by CENTER/ADVICE is apparent.

CENTER for Semiconductor Technology Optimization

The other major area of application of CENTER is in semiconductor device technology optimization. For

Table III. Results of Repeater Circuit Optimization

Case	Objective/Constraint	Bandwidth (MHz)	Gain (dB)	Ripple (dB)	CPU (hours)
Initial	–	347	75.4	0	–
Simulated annealing	Maximum bandwidth; gain >70 dB	670	72.9	6.6	0.5
WATOPT (a)	Also gain (550) <gain (mid-band)	644	72.8	2.9	2.0
WATOPT (b)	Also ripple <1	632	73.3	1.0	0.3

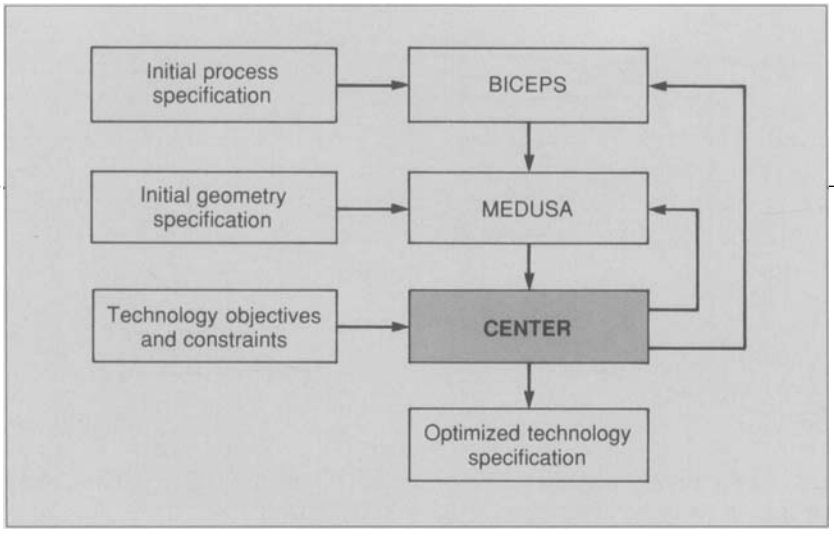


Figure 6. Optimization in the MECCA system.

this type of problem, CENTER has been integrated into the MECCA system.[20] The input to MECCA is a semiconductor technology specification, which includes design variables such as transistor lengths and control settings for the machinery used in IC manufacturing. MECCA produces an optimized technology specification, in which the design variables are set to values that yield the highest performance semiconductor devices. A typical design goal of MECCA is to maximize both device gain and current drive capability.

90

Figure 6 is a diagram of the MECCA system. The simulation of the design performances in MECCA is a two-step process. First, the BICEPS[21] process simulator generates doping profiles from the technology specification. Doping profiles specify the physical structure and chemical composition of a semiconductor device, from which its performance can be determined. Second, the MEDUSA[22] device simulator quantifies device performance using the doping profiles. The MECCA system has been used for parts of the process and device design for a 1.25-μm CMOS technology. Results of optimization of an *n*-channel MOSFET are shown in Table IV. The design goal is to maximize transistor gain and current drive capability while keeping the leakage current below a specified value. The design variables are the oxide thickness and channel length of the transistor and two impurity implantation doses of the manufacturing process. The changes in design variables and performances quoted in Table IV are with respect to a good initial design estimate provided by semiconductor processing technologists. They show that substantial changes in the values of the design variables may be needed to improve designs.

Conclusion

In this paper, we describe the functional capabilities and unique architecture of the CENTER generalized design optimization system. The functionality is directed

Table IV. Results of *n*-Channel MOSFET Technology Optimization

Design variable	Design objective/constraint	Specification	Optimization result
–	Gain	Maximize	Up 120%
–	Current drive capability	Maximize	Up 95%
–	Leakage current	< 1 pA/μm	Satisfied
Oxide thickness	–	> 150 Å	Down 25%
Channel length	–	> 0.5 μm	Down 25%
Tub dose	–	$> 2 \times 10^{12}$	Down 30%
Threshold dose	–	$> 2 \times 10^{11}$	Down 60%

NOTE: pA = picoampere; Å = angstrom.

toward IC design optimization, via the CENTER/ADVICE system, and semiconductor technology optimization, via the MECCA system. The breadth of capabilities also makes it useful for general design optimization, statistical analysis, and robust design. Ease of integration of simulation codes is a major strength of the architecture, as is ease of integration of new task modules. CENTER is a dynamic system to which new, state-of-the-art optimization and statistical design software will continue to be added.

Perhaps the greatest strength of CENTER is its evolution as an integrated system. The ability to create analysis and design strategies using the modules available within CENTER magnifies the power of each individual module. The SAO strategy (which uses sampling, approximation, and optimization modules) exemplifies this cohesiveness.

Acknowledgments

Many colleagues have contributed to CENTER concepts, software, and applications. In particular, we thank Chip Brewster, Ta-Fang Fang, David Gay, George Howlett, Kun-Chen Hsu, David Lee, Janet Lentz, Sally Liu, Shahriar Moinian, Jim Prendergast, Kumud Singhal, Prasad Subramanian, and Margaret Wright for their participation in this project.

We also thank Rick Becker for his valuable comments on a preliminary version of this article.

References

1. R. K. Brayton, G. D. Hachtel, and A. L. Sangiovanni-Vincentelli, "A Survey of Optimization Techniques for Integrated Circuit Design," *Proceedings, IEEE*, Vol. 69, No. 10, October 1981, pp. 1334-1362.
2. W. T. Nye et al., "DELIGHT.SPICE: An Optimization-based System for the Design of Integrated Circuits," *IEEE Transactions, CAD*, Vol. CAD-7, No. 4, April 1988, pp. 501-519.
3. A. V. Aho, B. W. Kernighan, and P. J. Weinberger, *The AWK Programming Language*, Addison-Wesley, Reading, Massachusetts, 1988.
4. R. Chadha et al., "WATOPT—An Optimizer for Circuit Applications," *IEEE Transactions, CAD*, Vol. CAD-6, No. 3, May 1987, pp. 472-479.
5. W. H. Press et al., *Numerical Recipes, The Art of Scientific Computing*, Cambridge University Press, Cambridge, England, 1986.
6. A. Osyczka, *Multicriterion Optimization in Engineering*, Ellis Horwood, Chichester, England, 1984.
7. B. A. Murtagh and M. A. Saunders, "MINOS User's Guide," *Stanford University Technical Report*, SOL 83-20R.
8. D. M. Gay, "ALGORITHM 611—Subroutines for Unconstrained Minimization Using a Model/Trust-Region Approach," *ACM Transactions, Mathematical Software*, Association for Computing Machinery, Vol. 9, 1983, pp. 503-524.
9. J. E. Dennis, D. M. Gay, and R. E. Welsch, "An Adaptive Nonlinear Least Squares Algorithm," *ACM Transactions, Mathematical Software*, Association for Computing Machinery, Vol. 7, 1981, pp. 348-368 and 369-383.
10. S. Kirkpatrick, C. D. Gellat, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 22, 1983, pp. 671-680.
11. A. Hedayat and W. D. Wallis, "Hadamard Matrices and Their Applications," *Annals of Statistics*, Vol. 6, No. 6, 1978, pp. 1184-1238.
12. R. L. Iman and M. J. Shortencarier, "A FORTRAN77 Program and User's Guide for the Generation of Latin Hypercube and Random Samples for Use With Computer Models," *Sandia National Laboratories*, NUREG/CR-3624 SAND83-2365 RG, March 1984.
13. I. P. Schagen, "Internal Modelling of Objective Functions for Global Optimization," *Journal of Optimization Theory and Applications*, Vol. 51, No. 2, November 1986, pp. 345-353.
14. F. Severson and S. Simpkins, "Hadamard Analysis—An Effective, Systematic Tool for Worst-Case Circuit Analysis," *Proceedings, IEEE Custom Integrated Circuits Conference*, 1987, pp. 114-118.
15. L. W. Nagel, "ADVICE for Circuit Simulation," *Proceedings, 1980 International Symposium on Circuits and Systems*, Houston, Texas, April 1980.
16. S. R. Nassif, A. J. Strojwas, and S. W. Director, "FABRICS II—A Statistically Based IC Fabrication Process Simulator," *IEEE Transactions, CAD*, Vol. CAD-3, No. 1, January 1984, pp. 40-46.
17. L. W. Nagel and D. O. Pederson, "Simulation Program with Integrated Circuit Emphasis," *Proceedings, 16th Midwest Symposium on Circuit Theory*, Waterloo, Ontario, April 1973.
18. R. N. Kackar and A. C. Shoemaker, "Robust Design: A Cost-Effective Method for Improving Manufacturing Processes," *AT&T Technical Journal*, Vol. 65, Issue 2, March/April 1986, pp. 39-50.
19. *Selected Papers on Statistical Design of Integrated Circuits*, A. J. Strojwas, Ed., IEEE Press, New York, 1987.
20. E. J. Prendergast and P. Lloyd, "A Highly Automated Integrated Modeling System: MECCA," *Proceedings, Custom Integrated Circuits Conference*, Portland, Oregon, May 1985.
21. B. R. Penumalli, "A Comprehensive Two-Dimensional VLSI Process Simulation Program, BICEPS," *IEEE Transactions, ED*, Vol. ED-30, No. 9, September 1983, pp. 986-992.
22. W. L. Engl, R. Laur, and H. K. Dirks, "MEDUSA—A Simulator for Modular Circuits," *IEEE Transactions, CAD*, Vol. CAD-1, No. 2, April 1982, pp. 85-93.

Biographies (continued)

AT&T in 1987 and is working on the CENTER design optimization system, statistical optimization for robust design, and compact models for bipolar transistors. Mr. McAndrew has a B.E. in electrical engineering from Monash University, Melbourne, Australia, and an M.A.Sc. and a Ph.D. in systems design engineering from the University of Waterloo, Waterloo, Canada. Sani Nassif has been with AT&T since 1986 and is working on technology CAD, parameter extraction, and design for manufacture. Mr. Nassif has a B.S. in electrical engineering from the American University of Beirut, and an M.S. and a Ph.D. in electrical engineering from Carnegie-Mellon University. V. Visvanathan joined the company in 1984 and is developing software tools for accurate design verification of integrated circuits. Mr. Visvanathan has a B.Tech. in electrical engineering from the Indian Institute of Technology, Delhi, India, an M.S.E.E. from the University of Notre Dame, and a Ph.D. in electrical engineering and computer science from the University of California at Berkeley.

(Manuscript received January 20, 1989)