Analog design needs a change in perspective

By R.D. Middlebrook
Professor of Electrical Engineering
California Institute of Technology
Pasadena, Calif.

Analog design has long been overshadowed by digital because it is less the "real world," at least at the macro level, is analog rather than digital. Most modern electronic systems ultimately have to interface both input and output with the analog world.

Correspondingly, electronics students tend to think that analog design is more difficult than digital design because of the perception that analog problems are ill-defined and there is no unique answer. Indeed, analog designers commonly find themselves well-educated when they graduate but ill-equipped to handle the problems with which they are immediately faced in industry.

Whether you are an analog-design engineer, supervisor, or manager, you can significantly increase your productivity and value to your company, as well as gain the satisfaction of mastery over methods, by use of design-oriented analysis to obtain low-entropy expressions.

Over many years of teaching courses for audiences ranging from undergraduate students to experienced industrial engineers, I have discovered that there are many ways to help graduates accomplish the transition from student to engineer more effectively and efficiently. This process can be initiated by college teachers, and supported and augmented by engineering managers and supervisors.

The starting point is a change of perspective. Yes, analog-design problems are ill-defined in the sense that there are never enough equations to solve for the number of unknowns. In fact, not nearly enough, in the mathematician's sense. Still, as engineers, we have to solve the problem anyway. So, the usual negative approach installed in us at an early age: "I don't have enough information, so I can't solve the problem," must be replaced by the positive approach: "Somehow or other I have to find additional information to make the necessary trade-offs and approximations so I can do the design."

I want to expand on the following positive approach, expressed in terms of some new names:

- design is the reverse of analysis, so only design-oriented analysis is of any value.
- The result of design-oriented analysis is a low-entropy expression, from which more useful design is obtained than simply one numerical answer for an assumed set of component values.

Let's start with an examination of the familiar path facing a new graduate. Scenario: The graduating engineer falls off a cliff.

Typically, an electronics engineer graduates with his mind filled with formal analysis methods, theorems and derivations. He is well-skilled in solving simplified, sanitized analysis exercises that have unique answers: one answer is correct, all others are wrong. The system is this way for good reasons, namely that most such exercises are graded by teaching assistants who have neither the time nor the experience to evaluate an answer that doesn't fit the one provided by the instructor.

In any case, when the new graduate starts work, he is soon faced with a new situation, perhaps to prepare a response to an RFP (request for proposal). If you happened to you, properly, you led through a thickness document and eventually found a couple of pages worth of technical specifications. Upon absorbing this meager information, you were, like so many others, hit with the depressing realization that you had no idea where to start.

After a period of thrashing about, you may have sought the help of a colleague, who perhaps provided a blueprint for a previous similar design. With renewed optimism, you now had a circuit to work down all the equations you could think of. After two or three pages of rapidly expanding algebra, depression probably again set in as you realized that the algebra had become unmanageable and wouldn't lead to anything useful anyway.

After another period of thrashing about, which likely included a feeling of helplessness and the realization that nothing you had learned seemed to be of any use, you probably took yet another tack: you got your technician to breadboard the previous similar design so you could get a solid starting point. When the breadboard was fired up, you may have become distressed by a new realization: Your technician knew a lot more than you did, especially about how to debug.

Now began the final. and longest, phase of the new engineer's adjustment to the real world. You learned how design is "really done": by a combination of cut-and-try, knob-twiddling and guesswork. In time, these techniques grew and multiplied into competence and intuition, and you became an experienced design engineer.

The above scenario, of course, is oversimplified and exaggerated in order to make two points. First, design is the reverse of analysis: one starts with the answer, which is the specification, and one has to work back to what circuit configuration to begin with and what component values to use.

Second, what one is taught is mostly analysis, so there is a wrenching transition to the real world, like falling off a cliff. The engineer restarts his learning process from the empirical approach, while most of his academic knowledge, after his first abortive attempt to apply it, rusts away in his mind and is forgotten.

This situation is wasteful and inefficient, besides being unfair to the engineer. Despite well-advised and well-meaning efforts to increase the amount of "design" in engineering curricula, design remains an application problem following, and separate from, the conventional analysis.

I believe design can be integrated into analysis at a much more fundamental and detailed level.

"I believe design can be integrated into analysis at a much more fundamental and detailed level."

The above scenario, of course, is oversimplified and exaggerated in order to make two points. First, design is the reverse of analysis: one starts with the answer, which is the specification, and one has to work back to what circuit configuration to begin with and what component values to use.

Second, what one is taught is mostly analysis, so there is a wrenching transition to the real world, like falling off a cliff. The engineer restarts his learning process from the empirical approach, while most of his academic knowledge, after his first abortive attempt to apply it, rusts away in his mind and is forgotten.

This situation is wasteful and inefficient, besides being unfair to the engineer. Despite well-advised and well-meaning efforts to increase the amount of "design" in engineering curricula, design remains an application problem following, and separate from, the conventional analysis.

I believe design can be integrated into analysis at a much more fundamental and detailed level.

"Design" is a process that follows a sequence of steps. One starts with a simple, approximate model, perhaps a block diagram, and some basic quantitative relationships that establish the required functions and number of stages. One then gradually augments the model with more detail, progressing toward the right as the accuracy/simplicity trade-off shifts toward greater accuracy and less simplicity. Iterations are necessary when a tentative choice has to be changed, or when simulations or experimental measurements do not agree with predictions.

Time and cost are important ingredients in the trade-offs, especially in determination of the termination point. An efficient design process requires a smooth and gradual progress. It is especially important not to resort to the computer too soon, and to resist the tendency to expect the computer to do your thinking for you by giving it problems to solve at once. The computer can actually introduce additional iterations, unless the user is thoroughly familiar with the algorithms embodied in the software.

Each design iteration loop can be considered as a local feedback process. The analysis result is compared with the specification, which is the desired answer, and discrepancies are to be corrected by modifying the model and/or changing numerical values. To do this, the analysis must be worked backward or forward in terms of the contributions from the original circuit elements.

Working the iteration feedback loop is the essence of the design process, and is facilitated if the initial result is obtained by design-oriented analysis, which is simply a name to emphasize that analysis must be usable "backward" for design, and any analysis that cannot be thus used is a waste of time.

Well, what is design-oriented analysis? It is the process of controlling and guiding the algebra so that the result is a low-entropy expression, defined as one in which terms are ordered, or grouped, so that additional insight is obtained into the relative importance of the various contributions to the result. This is the source of most information needed for the design, and substitutes for the missing equations that would be needed to solve formally for the number of unknowns.

The word "entropy" is borrowed from physics, and defined here only qualitatively, as above. In contrast, a high-entropy expression is one obtained by "blind" application of algebraic manipulations, usually leading to sums of products of circuit elements, and gives no insight into how the relative values affect the result.

Lowering entropy requires input of en...

Continued on page T30

Dr. R.D. Middlebrook is professor of electrical engineering at the California Institute of Technology (CalTech). His publications include numerous papers, a book on solid-state device theory, and another on differential amplifiers. The recipient of numerous awards, Middlebrook in 1991 was given the Ewing Marion Kauffman Medal of the Franklin Institute.
What is analog design?

Continued from page 75

nergy; in this case, it takes mental energy to
direct algebraic manipulation to derive the
low-entropy result from the high-entropy
one. It would be much more efficient, and
easier, if one could derive the low-entropy
result directly, without letting the entropy
rise in the first place. This is where the
various methods of design-oriented analy-
sis come in, one of which is use of Thiven-
án's and Norton's theorems.

Still useful

These theorems taught in school are useful
in that every time you use Thivenán's or
Norton's theorem, you get rid of one loop
or one node of the circuit. Therefore,
by successive use of the theorems, you can
reduce a complicated circuit to a single loop
and write the result by inspection. And,
there's a bonus: Not only do you get the
result with less algebra, but the result also
is automatically in low-entropy form,
because the elements automatically get
grouped during the reduction process.
There are many more sophisticated
techniques (read, tricks, shortcuts) for us-
ing design-oriented analysis to obtain low-
entropy expressions that are useful for
design. It's just a matter of practice—and
motivation. Performing design-oriented
analysis to produce low-entropy expres-
sions does not require learning new prin-
ciples or theorems. If anything, it requires
unlearning some things that have been
deeply ingrained from an early age, such
as: If you don't have as many equations as
unknowns, you can't solve the problem—and
that you shouldn't make an approxima-
tion if you can't justify it.

On the contrary, a design engineer has
to solve the problem with insufficient
equations, and one of the ways to do it is to make all the
approximations you can, justified or not,
leave behind a wake of assumptions and
approximations, so at least you get an an-
swer—which is better than no answer at all.
Of course, you have to go back and check
them out. The best that can happen is that
the approximations are OK, in which case
you're home free; the worst that can happen
is you have to go back and reject some of the
approximations. Regardless, you can't lose by
trying—even a failed approximation usually
suggests an alternative.

What design-oriented analysis does
need is development of a different per-
spective on the goal of analysis: You con-
trol the algebra, rather than let it control
you. More the answer come out in low-
entropy form.

Uncertain principles

One might say that we all have to go
through a process of "technical therapy": a
sort of Freudian regression in which we
have to unlearn some of the earliest things
we were taught so that we can make a new
start in which algebra is viewed as our
servant, not our master; and that approxi-
mations are valuable, even essential, in
getting an answer, not something to be
feearfully concealed in a rearguard ac-
tion against overwhelming algebra.
In general, "technical mental health" is
achieved when we adopt the overall positive
viewpoint that the problem can indeed be
solved with the proper approximations and
assumptions. We, as academics, can re-
semble the quandaries of our students;
the engineers are better prepared to survive the transition
to design engineer, thus giving them a chance to benefit from their formal background.
Supervisors and managers, who may or may not have had design experience,
can also benefit from encouraging their de-
sign engineers to develop low-entropy
results. Typically, reports and design re-
views are presented with results in high-
entropy form, analytic expressions that
give no information other than that ob-
tained by substitution of a set of element
values. The design engineer says some-
thing like, "It seems to be working, at
least over part of the range." You, as a supervisor or manager, can only say, "Well, looks as though it's
coming along all right, carry on." In con-
trast, if you insist that results be presented
in low-entropy form, you not only ensure
that your design engineer has better insight
into and control over the design, but you
yourself can interpret the results and offer
suggestions and guidance.

SOURCE CODE

By Ray Weiss

An analog hero

I've spent a career dodging analog electron-
ics. It started in school when I ran into the
underlying indeterminism that characterizes
the analog world. In analog, you do a pre-
liminary circuit, get your equations together,
and start "guessimating" the component
values. I headed for more deterministic dis-
ciplines, such as high-level mathematics,
hardware logic/systems design and program-
ing. These avoid
the underlying real world which, at best, is messy. And
lots of others did the same. So, now there's an analog design-
er shortage.

Over time, I became adept at evading analog—even as
clock rates ominously rose, and digital signals increasingly re-
sembled RF reflections. That ended a month ago, when I ran
into R.D. Middlebrook, an eminent Caltech professor and ana-
log researcher. A soft-spoken, transplanted Briton, Dr. Mid-
dlebrook is a polite-but-passionate analog zealot. And he is out
to make analog design easy to learn and do.

He's also persuasive. So persuasive that I found myself—
me, a digital and software guy—actually sitting in on an analog
design class for industry. Even more surprising, I understood
it! For the first time, I could get to solutions without develop-
ing a hermia from heavy algebraic lifting.

And it wasn't just me. One of the top power-supply consul-
tants in the country also was there. "I use a lot of these tech-
niques already," he said. "And some of these I've never heard
of, but I'm going to be using them. They really simplify design."

A number of the engineers at the class are ex-Middlebrook
acolytes who came back for recharging. One, Mark Fortunato,
now at Citicorp/TTI (Santa Monica, Calif.), said, "I use his tech-
niques all the time. I'm back because I couldn't read all my old
notes and there are even more techniques now."

The class was a real eye-opener. Here's a peek at some of
Middlebrook's magic shortcuts, tricks and techniques for how to:
• Build low-entropy (simplified, low-complexity) equations
for easy design solution.
• Use an iterative design process or methodology—starting
from initial analysis cuts to a final design—minimizing algebraic
manipulation and maximizing design choices.
• Solve quadratics without square roots—simple approxi-
mations to reduce hard-to-work-with equations.
• Reduce complex linear circuits to easily analyzable forms
using Thivenán's and Norton's theorems, and how to do alge-
bra on the circuits, instead of working with large, cumber-
some equations.
• Draw/find circuit poles and zeros without heavy algebraic
lifting, using lapped poles and zeros, and neat estimation
techniques on loglog drawings (db or phase vs. freq).
• Add an additional component to the circuit for analysis with-
out having to start over (Additional Element Theorem).
• Find loop gain by injecting a test signal into a closed loop
instead of breaking loop feedback path and calculating A and K
separately (Feedback Theorem).
• Find I/O impedances from circuit gain by taking simple
limits, reducing to one equation (I/O Impedance Theorem).

If you're an analog designer and aren't using these short-
cuts, you're working way too hard. There's a better way.
Middlebrook calls it "design-oriented analysis." As he puts it:
"Most analog design works backward from a solution. Unlike
school problems, you typically know the answer—what you
need is a design. The object is to do design. So, set up analy-
sis criteria that make it easy to write and analyze equations.
Reduce equations so you can easily see the relative impor-
tance of elements, as well as select key values."

It's a streamlined way to do analog design, eliminating equa-
tional overload. By simplifying circuits, by making viable approxi-
mations and by using easy representations, equations reduce to
intuitive forms; you can see the circuit and select values.

Unfortunately, analog design isn't going away; in fact, it's
going critical, as clock rates climb. Like it or not, we will have
to deal with analog beast. And so I'd like to offer a toast to
Dr. R.D. Middlebrook—he's making analog easy.

88 December 17, 1990 Electronic Engineering Times