



Jim Williams is a staff scientist at Linear Technology Corp. Longtime EDN readers recognize Williams as a vital contributor of analog-themed articles over the last 30 years. Williams has worked for 27 years at Linear Tech and has held previous roles at National Semiconductor and the Massachusetts Institute of Technology. EDN Technical Editor Paul Rako asked Williams about the changing role of application engineers—from essentially a support function of salespeople to full-blown system designers. Williams explains how a modern view of application engineering can ensure that your company will remain innovative.

## Application engineers: serving *the* customer

*Jim Williams  
on the  
changing role  
of application  
engineering*

**Application engineers have a different job today from in the old days. How have things changed in the years you've been in the business?**

I'll direct my comments at analog-application engineers. The big change is that the customer who calls you up or otherwise contacts you is generally not an analog designer. They knew a lot more 20 years ago about what they were doing than you ever would. They had specific questions on specific line items in a data sheet.

Those people are still out there, but they're in the minority now. The customer who's calling you up today, as a rule, is not an analog designer but needs to access analog technology. So rather than a specific question about a specific part or a specific characteristic of a specific part, they'll call you up and they'll tell you, "I have so much space, so much time, so much power, and so much money. What do I do?" They're not asking you detailed technical questions. They're asking you what to do.

That means you're in the service business now; you're providing design services. That also means

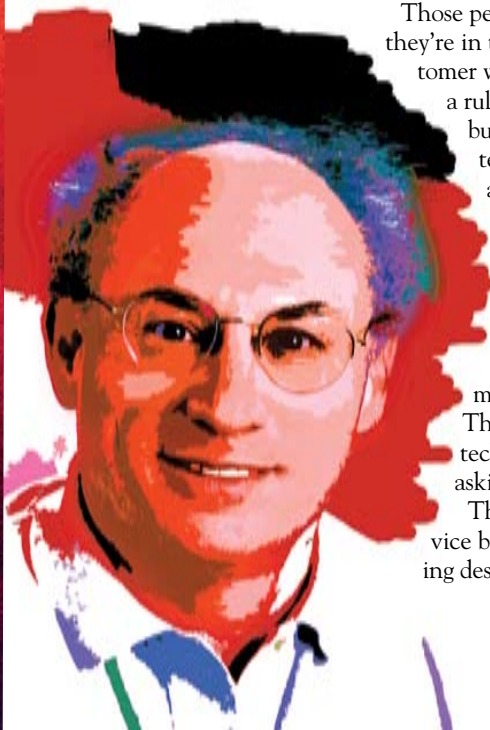
that the typical analog customer is not looking at your parts catalog. They're coming to you to solve a problem. They're ignorant of analog technology, but that's no crime. They've got 500 line items on their board to check off. They're not analog designers. They just need to access analog technology, and that's what you're going to do for them. They're not coming to you to ask you questions about components. They're coming to you to solve a problem.

**So 20 years ago, what was an application engineer doing?**

The classic application-engineering job 20 years ago was [working on] data-sheet support and writing app notes about existing parts. ... [They provided] high-level support for existing parts—how to use existing parts—in the applications they [felt they were] likely to be used in. So it was largely a reactive job.

**And now what do you see?**

That reactive component is still there, although in this company, the designer writes the data sheet; we insist on that. He may seek assistance with the application engineer, but the designer is responsible for getting the data sheet out the door. Those components that application people were engaged in 20



## **“The fundamental responsibility of an analog-application engineer is to understand his customer’s problem and provide a solution.”**

years ago are still there. But in a well-run analog company, they’re second- or third-tier parts of the job. The fundamental responsibility of an analog-application engineer is to understand his customer’s problem and provide a solution.

For me, that solution may involve this company’s products. I’ve also provided solutions to customers that use this company’s products and competitors’ products because those are the realistic solutions. I’m the customer advocate, and I’m seeking a long-term relationship with a customer. But what I’m doing and what I hope the other application engineers are doing around this company is ... servicing those customers who want to access analog technology but don’t have the expertise to do it and don’t have the time to develop that expertise.

That’s a large part of the job. Part and parcel of it is that you’re garnering a feeling for what needs to be built next. You’re defining new products. And an application note today that’s written by an application engineer may involve support of an existing product, but it’s much more likely to be written around current technical issues.

### **Do you distinguish between an application note written by an application guy versus the application section of a data sheet written by a design engineer?**

There’s a complete difference. An application note, if it’s properly written, stands a good chance of having a 10- or 15-year lifetime because it’s issue-centered; it’s not product-centered. There’s product in it that is illustrating various types of solutions, but, if the application note is really well-written and well-thought-out, it’s applicable even when the parts are long gone from the scene. You’re writing about issues. You’re writing about approaches.

### **Do you make a distinction between a factory-application engineer and**

### **a field-application engineer? Are those two different jobs, or are those roles converging?**

They’re two different jobs, but there’s overlap. The biggest single difference between the two is that the factory-application guy is much more future-oriented, and he’s much more laboratory-based because he needs a laboratory. The field guy, unless he’s got a laboratory at home, hasn’t got access to a laboratory, or, if he does, it probably isn’t as well-facilitated as [that of] the factory guy. So the factory guy can spend a lot more time on research and development for future products. The field guy is busy servicing existing customers with existing problems, which is important, obviously, ... but the factory guy’s not doing that. If he’s focused on servicing existing customers with existing products and getting existing sockets, he’s mortgaging his future. He’s eating his seed corn and developing nothing.

That’s the fundamental difference between the two positions: The factory guy has the luxury of being able to spend more time on futures and on issues. If factory engineers are pushed toward devoting 90% of their effort to getting sockets, you’ve successfully mortgaged the company’s future. Whereas the field guy is with customers all the time. They’ve got today’s problems surrounded by today’s parts that they can get their hands on now. So they’re the heroes of today. But in a well-run application effort, the heroes of tomorrow are in the factory. That doesn’t mean the factory doesn’t provide backup for the field guy when things get sticky. You’ve got to do that, but that plays into tomorrow, because you look at what comes over the wall that the field people can’t handle, and you see trends, which suggest products.

### **And it doesn’t physically mean that the guy works at a factory. It could be that he works at a local office but is acting as a factory-application guy, right?**

We’re not saying he has to be at the headquarters. ... He needs a lab, and he needs time. The greatest leverage any engineer has in doing his job is time. And, nominally, the field guy is spending the bulk of his time working with customers on existing problems with existing products. The factory guy should be spending the bulk of his time thinking about what needs doing in the general sense.

### **Do the application groups for field and factory have a responsibility to show customers how to use software tools?**

Well, let’s talk about tools. There’s LT Spice. There’s Webench. There are all the various software tools. But there are also screwdrivers, shears, curve tracers, and X-acto knives. The application engineer’s job is to emphasize and show the customer what tool is appropriate for what task. For some tasks, LT Spice is the appropriate tool. For other tasks, a cut-down X-acto knife is the appropriate tool, and that should be reflected in applications. What’s the real down-in-the-dirt way to get from A to Z? Is it Spice? Is it an X-acto knife? Is it a breadboard? Is it cutting copper clad? Is it some fusion of all of those? Tool development, tool use, writing it up, how to measure, how to simulate, when to measure, when to simulate, where to simulate, and when to cut copper clad are all part of the application engineer’s job in educating the customers on how to solve their problems.

Customers like data sheets. Customers like app notes. Customers like publications that educate them. Customers like advice over the phone. Customers like software programs that help them design. But what customers love, what sells a product like nothing else, is that simple little cardboard box arriving in the mail with a breadboard that works when they plug it into their system. No sales pitch, no sales routine, no software program, no phone conversation, no e-mail sells products like a working

**“There’s no patent on intelligence. Anybody can learn to do anything. It’s just people haven’t got the time.”**

breadboard mailed to a customer. I can’t say that loud enough or long enough. Nothing sells like a board that works in the customer’s system. That is the ultimate analog-application support. Nothing beats that.

**What about compensation of factory- and field-application engineers? The field people get bonuses for filling sockets. Are you against having factory-app people compensated by filling a socket?**

This might get me into trouble in some sectors around here, but I’m against all forms of goal-setting for factory-application engineers because they’ll pursue those goals and let other things slide by the wayside that could be the future success of the company.

If you sit down and you agree with somebody that you’re going to be rewarded if you do this, this, and this, then that’s what a lot of people are going to be tempted to do. Then they’re going to see this, this, and this, which look interesting and, potentially, fruitful, but they know if they do this, this, and this, they’ll get rewarded.

It may be that goal-setting in a field engineer’s job seems unavoidable, because they’re servicing existing customers with existing parts, and that’s the only metric that management can use. But in a factory job, which is nominally R&D-based, if you line up a bunch of goals for somebody and tell them that’s the way to success in this company, you’re destroying the company’s future. You’ll have a bunch of people who are successful according to their goals and a company that stalls in innovation.

**I’m curious how you see application engineering helping with reference designs in general.**

There are two ways applications can export design expertise—which, ostensibly, I hope they have—into the world. One way is what I call “the noble way.” That’s through application notes that

are topic-centered. The thesis there is that an educated customer is a better customer, and an educated customer will come back for more, I hope, to your company.

The noble way says you educate the customer through publication. The less noble but effective way is the reference design where the customer says, “I haven’t got time to come up to speed. You’re providing me with these publications, but I haven’t got time to come up to speed. I need that little cardboard box with the breadboard.”

That is a reference design. It’s essential because customers are out of time. They haven’t got the time to come up to speed to execute the architectural issues in a circuit themselves. And there’s also a real marketing issue. Through-hole is dead. We can breadboard here at the factory, but, by and large, customers can’t try stuff anymore. The parts are too damned small.

You’re going to have to take the part you’re trying to sell for the problem you’re trying to solve, and you’re going to have to incarnate it on a board-level reference design, because the customers haven’t got the time to develop the expertise they need to execute a design and because they have a hard time breadboarding and playing.

Reference designs and demo boards are important. I would say of the demo boards that get adopted and the reference designs that get adopted, a third get used pretty much the way they are, and two-thirds get used as a place to start. The customer comes back to you and says, “I fired up your demo board and looked at your reference design. It seems to work pretty well, but I needed twists here, there, and the other place.” But the point is, the reference design provides an advanced place to start talking from.

There’s something else that comes to mind with reference designs. In most analog companies 20 years ago, the word from on high was “Don’t get involved in your customer’s design.

Don’t accept responsibility for the design; there are potential legal problems.” Many companies went out of their way to tell their application guys, “Don’t get knee-deep in a design and accept responsibility.” Today, accepting responsibility is a sales tool. That’s turned 180°. There is a range of solutions available from a number of competitors, and, if you want to distance yourself from the competition, you’re going to have to accept responsibility. It’s based on the complexity of the product, and it’s based on its availability in one form or another.

**So this way is the new way. That’s how you see application engineering developing?**

The analog business is a service business. I can walk into a customer’s facility and hand them data sheets and app notes and parts. That may have been a sale 25 years ago. It isn’t anymore. They haven’t got the time to read the data sheet. They haven’t got the time to develop analog expertise. They may not have the inclination; they’ve got other things to do, and they can’t breadboard with the parts. They say, “So your parts are interesting, your app notes are pretty, your data sheets are pretty, but I need something I can clip into my system that’ll work.” That’s a complete shift from 25 years ago, when management was telling application engineers, “Be very careful about getting knee-deep in your customers’ designs. We don’t want to get sued for a field recall.” That’s 180° out of phase with what’s going on today. You’re ... over your knees, over your head in your customers’ designs.

**With many companies, there’s no analog team. There’s no analog engineer.**

They’ve got other things to do. There’s nothing magic about the analog field. It’s just people have other things to do. There’s no patent on intelligence. Anybody can learn to do anything. It’s just

**“If you can’t explain how something you did works to a general audience in a general kind of way, you don’t know how it works.”**

people haven’t got the time. You walk into a customer’s facility, and you look at their blackboard, if they have one; there’s 50 items to be checked off there, and four of them are analog.

They don’t need to develop in-house analog expertise. They’ve got analog companies who will do it for them. That doesn’t mean there aren’t companies out there that are well-steeped in analog expertise. It just means that most of the products being built today that use analog technology are being built by companies that are not steeped in analog-design techniques and don’t have to be. That’s what they look to us for.

### **Where do we find this new breed of application engineers?**

The far side of Alpha Centauri. I don’t know where the hell you find them. You like to think you can breed them, but it takes a long, long time and a lot of burning of fingertips. To some extent, you can breed them. To some extent, they find themselves. You still find kids coming out of college who’ve been playing with electronics since they were in grade school. Those people still exist. They’re aberrant. They’re weirdoes, but they’re wonderful weirdoes.

You still find those people who got addicted early and found a way, despite the surface-mount revolution, to hack electronics in grade school and high school. You do find people who’ve been doing board-level design, sometimes your own customer, who wants to come over—not often, but it happens. Those are the two major sources: lifelong circuit freaks and system guys who’ve been working for board-level-product houses who want to come over and do this kind of work in a semiconductor company. Also, to some extent, inbreeding within the company [and] mentoring [help]. But there is no official, if you will, path toward finding these people. It’s quantum mechanics. If you line up enough people on one side of the fence, experienced system designers who want to work in a semiconductor company

will appear on the other side of the fence. No one knows quite how. It’s a tunneling process.

### **Do you think application engineers should be writing magazine articles as you do?**

Application engineers should definitely write articles, for a couple of reasons. The most obvious reason is that it’s good for the company’s image, but, more important, if you can’t explain how something you did works to a general audience in a general kind of way, you don’t know how it works. It’s a great way to test your own level of understanding of what you just finished doing.

Writing is essential. It’s an essential part of an application engineer’s job and communication skills. It’s harder to find people with really good communications skills than with technical skills. Writing is important because it tests your knowledge of your ability to understand what you really did. It puts your company in a good light, and, career-wise, it puts you in a good light. I can’t imagine working on a difficult problem over a protracted period of time and then not writing it up. Make your contribution. If you’ve got something you think is worth talking about, make your contribution.

### **I see a difference between an artist and a tradesman. An artist will say, “Here’s the palette, here’s the paint I used, and there’s my canvas. Have at it.” They share all that important stuff, whereas a tradesman would say, “Oh, this is my secret little thing, and I keep it to myself.” You take kind of a higher-plane view of things and act like an artist. Why not keep everything secret?**

I think [the “father of the atomic bomb,” Robert] Oppenheimer was right. There are no secrets. The only secret to the atomic bomb is that it works.

Once a skilled technocrat sees that

something can be done, the only question is how. There’s no advantage to secrecy. Whatever advantage you’ll get by protecting some body of knowledge that you have is dwarfed by the goodwill and the good orders and the sense that you’re a problem-solving ally when you disseminate that knowledge.

### **So customers at system companies appreciate the openness of a modern application engineer?**

They’re looking for somebody they can trust technically as a partner. Most technological trade secrets are short-lived at best. You’ll do your company, your professional reputation, your profession, and everybody else a lot more good by saying, “Here’s this problem I had, here’s how I solved it, and here are the results.”

There’s really no advantage to holding back. I don’t recall ever holding back a measurement or design technique from publication in an app note. Remember a number of years ago, when there were soft errors in memories? Intel figured it out. It was [caused by] alpha particles. Intel released it and gave it away, and a lot of people said, “Intel is crazy. Intel shouldn’t do that.” So you’re making friends, you’re generating credibility for yourself and your company, and [disseminating it obviates any] short-term benefit that you can gain by keeping things close to the chest.

There are certain manufacturing processes, trade secrets, and the fab, stuff like that, that you’re going to play close to the vest. But you’re not going to have to play them close to the vest for long because they’re going to be obsolete in six months or a year anyway. But in application engineering, measurement technique, circuit-design technique, whatever, you’ll generate more new customers by printing it all in an app note than you’ll lose because some competitor took your scheme and ran with it.—interview conducted and edited by Paul Rako