

Where should your systems grow?



For years, the semiconductor industry has been chasing the Holy Grail. Almost from the beginning, the industry objective has been to integrate more and more transistors onto smaller and smaller ICs. Today, silicon

vendors have the capability and skills to put millions of transistors onto tiny chips. Their efforts have spawned a new breed of company that aims to relieve you of having to redesign standardized and common functions. EDA vendors have also developed tools meant to help you design million-transistor circuits and integrate your circuits with function blocks acquired elsewhere to fill these multimillion-gate-capacity ICs. All of these companies would like you to believe we're entering a brave new world of systems on silicon.

Now, I'm a bit of a cynic, so when I hear everybody talking about systems on silicon, sirens go off, and red flags go up. I start wondering if we've got an out-of-control bandwagon and if we've abandoned objectivity and reason. So let's look carefully at the situation.

Designing a multimillion-transistor ASIC requires time and money. Rather than designing at the gate level, engineers often design these minisystems using HDLs and function cores that speed the process. That practice is good. Like programming in C instead of using assembly, working at a higher level of abstraction increases efficiency and helps give engineers the productivity they need to design million-transistor circuits in months instead of years.

Then, there are the economics of ASICs. How many designs—of any size—are done correctly without any re-spins? At more than \$200,000 per mask set, corrections can be expensive—even before you factor in the time it takes to create the masks and process the silicon. And when you have relatively small volumes across

which you amortize the costs, re-spins can be prohibitively expensive—even when you don't need to redo the entire mask set.

In addition, fabs are expensive and need to keep pushing silicon through the lines. To do so, manufacturers generally want assurances of high production volumes. There are two key ways to ensure high production: Promise to buy lots of chips, or be willing to share your design with a bunch of companies that together offer the promise to buy lots of chips. Sharing, because it gives away your proprietary design, isn't such a good idea. There aren't many multimillion-transistor, one-manufacturer designs that ship enough pieces to keep the foundries full. Increasingly, system houses design the chips and license

them to the silicon vendors on the condition that the silicon vendors will give them a head start in selling the products. Though they make their proprietary designs available to their competitors, the system

companies hope to push technology fast enough to make that design obsolete by the time that head start expires. Playing this game is like walking a dangerous tightrope. If the system folks stumble, they face lower cost competitors that they've bootstrapped with their technology. And, if they succeed in obsoleting their own products before their head start expires, the system companies will find fewer silicon foundries willing to offer another head start.

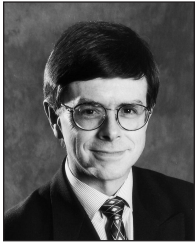
So, where does that leave systems on silicon? In my mind, it suggests that the million-unit designs need to be "not-so-big." The ICs should be big enough to take advantage of our ability to squeeze more transistors onto the silicon but not so big that the application focuses on an overly narrow niche. Instead, these almost-systems-on-a-chip should be such standard products as microprocessors, memories, and programmable logic that have broad use across thousands of applications and that you can customize quickly and at a low cost with jellybean logic or programmable parts.

When I hear everybody talking about systems on the silicon, sirens go off, and red flags go up.

MICHAEL C. MARKOWITZ
EDITOR IN CHIEF

Let me know what you think. Send me your comments via fax at 1-617-558-4470 or over the Internet at ednmarkowitz@mcimail.com or m.markowitz@cahners.com.

Cars and complexity



Today's typical automobile is more "environmentally friendly" than its predecessors of just a few years ago. Its growing electronics content plays a major part in such efficiency. Similarly,

engine control units run power plants at higher efficiencies and with greatly reduced emissions, and more efficient control of transmissions further boosts the mileage obtained from a tank of fuel. You can argue, too, that if GPS navigation systems become widespread, they will also reduce the negative environmental impact of cars by getting us to our destinations more directly—thus using less fuel.

All the same, we shouldn't minimise the contribution of all of the other engineering disciplines involved. Better mechanical design and aerodynamics contribute to efficiency, and chemistry plays a big part, too. Obviously better fuels, oils, coatings, and paints extend the efficient working life of a typical vehicle. After all, a significant part of a vehicle's overall environmental impact is the energy and raw material cost of building and maintaining it.

But there's a downside to all of this electronic "progress"; the newest car models are very complex, and although remarkably reliable, they do have a significant failure rate. Consider the following scenario: You own a car for a few years and are tens of thousands of kilometres into its lifespan. Although the car still runs efficiently and hasn't yet corroded, it suddenly develops an obscure and intermittent fault that your dealer has trouble pinning down. Somewhere in its dozens of micro-processors and sensors, something has gone wrong.

When dealing with electronics, many car mechanics have no ability (or inclination) to diagnose beyond the major subsystem level. Even worse, each of those subsystems, as replace-

ment parts, can be very expensive. Now, all of a sudden, you realise that your few-years-old vehicle, still in fine condition, is in fact economically unrepairable. It can only be scrapped and replaced, which is not at all an environmentally friendly outcome.

So, what are we designers to do? It turns out that our industry may already have a number of necessary solutions available in its bag of tricks. The problem is one of limited fault observability in a very complex system—rather like some of the problems that IC designers have been wrestling with for some time. It's one part of the thinking behind Hewlett-Packard's decision to create an entire division, based in Lyons, France, dedicated to serving the needs of the automotive industry. With it, HP plans to sell carmakers everything from LEDs to test gear, including test methodologies. Expect to see many of the same techniques that have evolved in IC test-

There's a downside to all of this electronic "progress."

ing: fault simulations run against vehicle models' subsystems (maybe even of the complete vehicle, but that's a tall order); fault dictionaries; and probabilistic analysis in which the mechanic gives the diagnosis system the fault symptoms and the system generates a list of the possible causes, thereby providing a degree of likelihood for each known symptom.

If you talk to automotive electronics designers, you'll soon realise that some of them believe that the solution lies in layering on yet more complexity—such as built-in self-testing or even redundancy and self-repairing mechanisms within electronic subsystems. As an increasing number of industrial, commercial, and domestic systems follow the path of escalating complexity, that sort of thinking may have to be applied much more widely. Or, to stand the argument on its head, there's nothing totally unique about chip design. We are entering an era in which very complex system design will be pervasive, and the techniques available to ensure that such design remains manageable are just starting to emerge.

GRAHAM PROPHET
EDITOR IN CHIEF

Let me know what you think. Send me your comments via fax at +44 118 935 1670 or over the Internet at graham.prophet@rbi.co.uk.

Technical Editor Brian Dipert's commentary on the Linux operating system (EDN, July 2, pg 40), in which he expressed how EDA vendors view Linux users, drew a big response from *EDN's* readers. Here are some of their comments.

Linux users are indeed a minority—right now. Various estimates put Linux at around 5% of the PC-user base, which is roughly a tie with the Macintosh for the No. 2 position. But the Linux-user base has doubled every year since 1994 and is expected to continue exponential growth for a while longer. Linux is at the crossover point where a long, slow incubation turns into a period of explosive growth. In fact, Linux already dominates the Internet-service-provider market and is in use in most MIS departments of companies of 50 employees or fewer. No other OS is experiencing similar growth today. Application vendors that ignore Linux are gambling with their futures.

And it isn't because Linux is free that users are installing it. They install it because it's the best OS in existence. Linux has become the OS of choice in mission-critical network applications, because it's such a bulletproof workhorse. The low price just makes it much more available; if corporate management balks, the ultimate users buy it out of their own pockets and install it anyway. Many top-level executives are unaware that their companies' internal networks depend on Linux.

Adopting Linux is by far the best way to minimize support problems, too. Linux's cooperative support via the Internet, by both commercial companies and newsgroup volunteers, is effective and fast. Bugs don't have a chance

to get entrenched in Linux. Because far more programmers are continuously working on Linux than even the richest vendor could afford, problems often get fixed within hours of discovery.

Linux does have soft spots, but they're being dealt with. Third-party application software is steadily appearing in native-Linux form; a few examples are Netscape, WordPerfect, Corel Draw, LinuxCAD, and several office application suites.

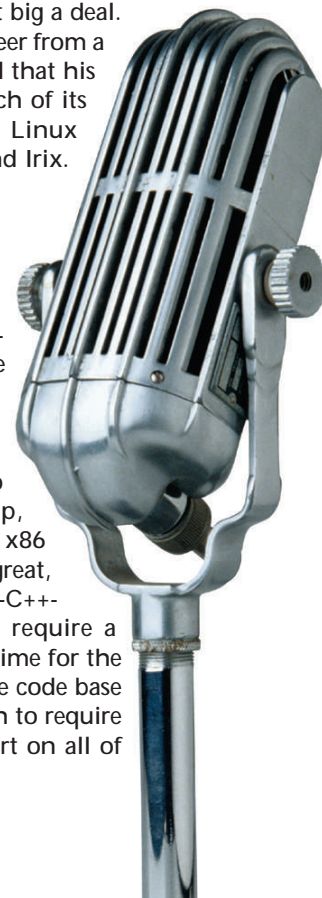
Jack Carroll

If a tool already exists for Unix, why have so few EDA vendors been willing to take the relatively minor step (compared with rewriting for Windows NT) of modifying it for Linux? In my experience, it's typically not that big a deal. I spoke last year to an engineer from a major vendor who confided that his company actually does much of its in-house development on Linux and then ports to Solaris and Irix.

Ian Board

*Raytheon Advanced Products
Torrance, California*

I'd like to contribute a thought from the perspective of an EDA-software developer. I have been interested in a way to port our Unix applications to a PC platform so we can demo them from a cheap laptop, which means using Solaris x86 or Linux. Linux would be great, except that GCC isn't ANSI-C++-compliant. To port would require a large amount of developer time for the port, and the changes to the code base would be significant enough to require a full quality-assurance effort on all of



ednmag.comment

the other platforms. On the other hand, once HP fixed its compiler, porting was relatively painless. For this reason alone, the choice between Solaris x86 and Linux is made for me—Solaris or nothing.

From my point of view, the lack of standard development tools increases the cost of a port. The more expensive the port, the more revenue I would have to have contingent on a Linux version before I allocate scarce development resources to it.

Jason McCampbell

The biggest question Corel Computer had to answer in deciding to support Linux was, "How can Corel run a business if it gives its software away for free?" But promoting Open Source Software in no way impedes Corel Computer's business objectives of becoming a highly profitable organization. The selection of Linux was based on the performance of the OS and the availability of the source code. Access to the source code has allowed Corel Computer to build a highly robust platform with numerous applications.

Oliver Bendzsa

I've been using Linux since kernel 0.99. I have a Linux installation in my home and a covert one at work. As for my home system, you're right about Linux users being cheap; I haven't spent 1 cent specifically on Linux software. As for work, I don't really care if the stuff is free or not. The most important thing is whether I can get my work done. When I used Windows 3.1, it crashed at least twice a day; Linux has crashed on me twice in three years.

In your statement that Linux drivers aren't available for various peripherals and that a vendor's tech support would have trouble helping a user debug a driver, you're missing the point. Linux users rarely ask for drivers from manufacturers. What we ask for is information so that we can develop drivers ourselves. If there's a bug in a driver and if the "common

people" have the source code, then someone will fix the bug and post a patch on the Internet. If the patch is wrong or is in some way inelegant, you can be sure that others will mention it.

Do I want EDA tools for Linux? Not necessarily. We have other "commercial" Unix workstations that work just fine for such things. Do I want EDA tools under some flavor of Windows? Definitely not. The few I've tried (like most Windows experiences) have been a frustrating series of crashes, lockups, and generally bad performance.

John Breen

I use Linux because it is technically excellent, fast, and stable. In other words, I use it for all the same reasons that any engineer would choose to use something superior over something inferior.

When was the last time you called the manufacturer for support on the software in the embedded microprocessor in your keyboard? I'll bet the answer is never. Why? Because the software in a keyboard actually works. You have to support only software that doesn't work. The solution to the support problem on any OS is to write quality software that actually does what it is supposed to do.

Robert E Canup II
Electronic Power Design
Houston Texas

The advantage of Linux is that you don't have ongoing costs for mandatory updates. I have four different versions of Linux running on four different computers, ranging from a 20-MHz 386SX to a 200-MHz Pentium. There's no need to upgrade or downgrade Linux to run it on a specific platform, and I've never had a crash.

The EDA vendors' argument about Linux users being a support nightmare is completely wrong. A typical Linux user has more people to talk to about a problem than a Windows user does. I don't mean just talking and

complaining, but to get real answers, so a vendor's hotline would be the last stop, anyway. And, yes, you can get a new version of Linux almost every month, but only fanatics update Linux that often. Usually you update when you need something that's not implemented in the version you have. For the computers in my company, this turns out to be about once a year. An update takes about one or two hours, including the restoration of all applications. Compare this to a reinstallation of Windows. You have to reinstall all your applications from scratch, and no backup is of any help unless you want to keep all the zombie entries in the startup files.

Ulrich Paul
Paul Elektronik
Leitershofen, Germany

EDA vendors think we're a minority, and so far we are. But Macintosh, Sun, HP, and SGI are minorities as well—and about the same size.

Petr Vicherek
Ericsson Mobile Networks

The point that Linux users are cheap is valid. Yes, we are. However, there are two observations to be added to this:

1. Windows/DOS is also popular, because applications are easily (if illegally) installed on multiple machines with only one license. Hence, many PC owners are cheap.
2. People will pay for tools to do their jobs if they feel they need to and the price seems worthwhile. Just because the OS is free, the applications don't need to be.

Andrew Hately
Eurocontrol

Many applications have been ported from Unix to NT but not to Linux, which makes no sense, since "porting" from Unix to Linux is almost only a recompile and link. Porting to NT is work!

Kenneth Scharf