

**P**owerful high-level software tools give domain experts in such diverse fields as aerospace engineering, medical electronics, mechatronics, and even graphics design increasing control over the implementation of embedded systems. Such software


packages can automatically generate code for target processors or FPGAs, leading to the possibility that dedicated hardware- and software-embedded-system designers may become members of an endangered species. The idea that domain experts might supplant electrical engineers gained wide currency during a 2008 Embedded Systems Conference panel discussion (**Reference 1**). A look at the current state of affairs suggests that domain experts and high-level-system architects—as opposed to coders and processor experts—have gained significant ability to participate throughout the embedded-system-design process. However, the need for C coders and hardware engineers is not going away. In most cases, domain experts and electrical engineers can better collaborate on quickly getting increasingly complex products to market.

ROBOT HAND BY SHADOW ROBOT

# HIGH-LEVEL SOFTWARE FOR EMBEDDED-SYSTEM DESIGN DOING YOUR

DOMAIN EXPERTS ARE GAINING MORE CONTROL OF EMBEDDED-SYSTEM DESIGN, BUT ELECTRONIC- AND CODE-DESIGN SKILLS REMAIN KEY TO SUCCESSFUL PROJECTS.

BY RICK NELSON • EDITOR-IN-CHIEF



**JOB?**

“The premise was that people who have to implement an embedded system and the people who know what needs to be implemented never know how to talk to each other,” says Jim Tung, a fellow at The MathWorks, commenting on the 2008 panel discussion. “Certainly, the world has moved away from that starting point. What you certainly do see now is collaboration. The people who have the implementation skills ... to deliver a device with a certain power consumption and performance profile can work more effectively with the domain experts—the people who know what the performance needs to be.” High-level languages, such as The MathWorks’ Matlab and Simulink, he says, enable that collaboration to take place.

## TEST SOFTWARE

National Instruments has been promoting its LabView graphical-design software, which initially served test-

### AT A GLANCE

✦ In most cases, domain experts and electrical engineers can better collaborate on quickly getting increasingly complex products to market.

✦ High-level languages, such as The MathWorks’ Matlab and Simulink, encourage collaboration between domain experts and engineers.

✦ National Instruments has moved into the embedded-system-design area, but the company is not moving away from test.

✦ Automatic code generation has reached the point at which the resulting code is efficient enough in memory and provides sufficient throughput to suffice for many applications.

and-measurement applications, as a tool for embedded-system design. “With the thousands of technical challenges the engineering community faces today in

applications such as medical, robotics, and ‘green,’ we don’t have enough embedded-engineering experts,” says an NI spokeswoman. “Therefore, the masses of domain experts out there become critical parts of helping out the engineering community. We’ve seen numerous domain experts be very successful using our graphical-system-design tools to design embedded systems in the medical, robotics, and renewable-energy industries, and we will continue creating embedded hardware and software tools that allow any domain experts and scientists to innovate themselves and design embedded systems.”

One NI customer that has successfully used NI tools in embedded-system design is Alliance Spacesystems, which makes mechatronics systems, such as robots for NASA (National Aeronautics and Space Administration). Shelley Gretlein, LabView real-time and embedded-product-marketing manager at NI, describes Alliance Spacesystems’ mechatronics group as a multidisciplinary team comprising aerospace, mechanical, electrical, and control engineers all cooperating on the same team. “You see much less pure EE [electrical-engineering] or strong embedded experience,” she says. “Much more, you see people coming from different backgrounds to build these systems.”

The move to the embedded-system area has been an evolutionary one for NI, but that doesn’t mean that the company is moving away from test. “Test is NI’s bread and butter,” says Todd Dobberstein, product manager of industrial and embedded technologies for NI. “We’ve been dealing with test and data acquisition since we started as a company.” The company’s test expertise, he says, complements the design capabilities inherent in LabView. At Alliance Spacesystems, NI’s Gretlein works with a group that began as a test customer and has increasingly adopted NI tools for design, leading to a unified tool chain.

PJ Tanzillo, biomedical-segment lead and embedded-software manager at NI, cites another customer that has applied NI’s tools in an embedded-system-design project: Sanarus successfully designed the Visica 2 cryoablation system for the treatment of tumors (Figure 1). Tanzillo describes the system engineer on the project, Jeff Stevens, not as a do-



Figure 1 The Visica 2 cryoablation system for the treatment of tumors (a) went from a requirements document to first revenue in 14 months, according to system engineer Jeff Stevens (courtesy Sanarus). Stevens used LabView to develop code for the CompactRIO platform (b) that served as the embedded engine in Visica 2 (courtesy National Instruments).

main expert but as an electrical engineer by training whose expertise is at the system-architecture level rather than the coding and hardware-optimization level. Such system architects may be able to apply their talent across multiple disciplines from robotics to medical electronics, but, Tanzillo says, they understand the underlying technology, albeit not necessarily down to the level of developing a driver for an ADC or writing HDL (hardware-description-language) code for an FPGA.

## DESIGNING THE VISICA 2

Stevens describes himself as a systems engineer with degrees in electrical engineering who thought that any code he had written would never see the light of day. That situation changed, however, when he joined Sanarus in November 2005 as principal system engineer. There, Stevens faced a tough deadline: developing a fully functional prototype of the Visica 2 within four months. “As the system architect, I could see that we’d need a custom PCB [printed-circuit board] for the microprocessor and all the I/O and that we’d have to outsource all the firmware because nobody at Sanarus spoke software,” he says. “That was the approach Sanarus had followed on its previous product, but that product was an order of magnitude simpler than Visica 2.” Further complicating the task was the fact that the team available to work on the project would be small. Besides Stevens, the team comprised a mechanical engineer; a “supertech,” who had a degree in psychology but was creative at prototyping, knew how to run milling machines and lathes, and could solder PCBs; and a project manager. “The joke was that we kept [the manager] out of the lab,” Stevens says. “I was thinking eight months might be enough to accommodate board and code spins plus integration, but four months fell squarely in the ‘no-way’ bucket.”

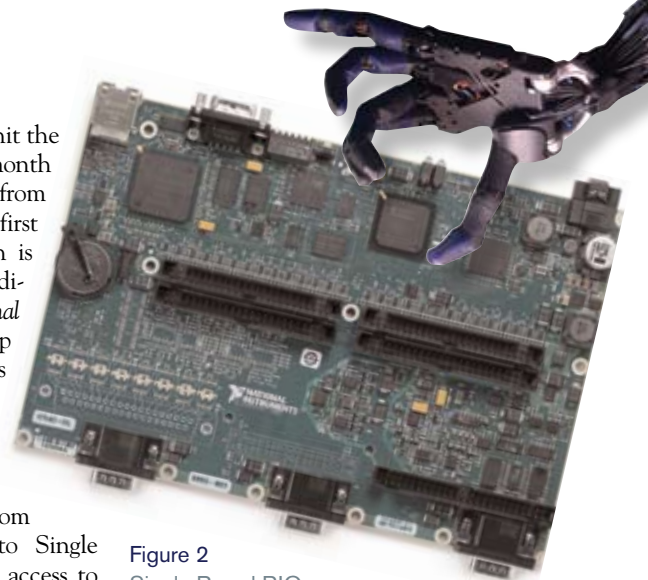
However, he then investigated NI’s CompactRIO (reconfigurable-input/output) platform as a hardware target for embedded code that could run software that he could develop on LabView—a tool he had heard of but had never used. Stevens presented his case to Sanarus management and got approval to adopt the NI approach. With the help of NI support personnel and a developer from Cal-

Bay Systems, he “comfortably hit the working prototype in the four-month target.” He adds, “Visica 2 went from a requirements document to first revenue in 14 months, which is pretty good for an invasive medical device. *The Wall Street Journal* awarded Visica 2 the runner-up prize for medical devices in its 2008 Technology Innovation competition.” (Reference 2). To support a lower-cost production version of the Visica 2, Stevens ported his code from the CompactRIO platform to Single Board RIO (Figure 2). “I got access to Single Board RIO before [NI] publicly announced it,” he says. “The porting took about 45 minutes.”

## HIGH-LEVEL ABSTRACTION

Sanarus completed the Visica 2 project without the intervention of traditional embedded-system designers. That situation may be the exception that proves the rule, however. “Think about the ‘good old days,’” says The MathWorks’ Tung. “If you wanted high-performance software running on your desktop, you had to write in assembler. That’s why Lotus 123 was so lean and so fast.” Subsequently, “optimizing compilers have reduced the need for people to work at that level, so they are able to work at a higher level of abstraction—say, C code,” he adds.

According to Tung, domain experts will want to work at an even higher level of abstraction with Simulink, for example. Automatic code generation has reached the point at which the resulting code is efficient enough in memory and provides sufficient throughput to suffice for many applications. “However, there are still times when you need to really tune and optimize that implementation, so the need in an embedded system for somebody to write in C code hasn’t gone away. It just happens less frequently,” he says. “If I am going to write something and really fine-tune it at the C-code level, I want to do that only once. If I’m going to handwrite something, then a high-level modeling language, such as Simulink, should be able to encapsulate it and make it available to the system-level engineers so they can look at the design-exploration trade-offs and make a much more informed decision, and



**Figure 2**  
Single Board RIO offers a cost-effective production alternative for CompactRIO prototypes. System engineer Jeff Stevens ported the Visica 2 code from CompactRIO to Single Board RIO in about 45 minutes (courtesy National Instruments).

[this approach] facilitates reuse.”

Software-development teams would continue to need to consider issues such as when to deploy an RTOS (real-time operating system) and when it would be necessary to consider race conditions and task overruns, Tung says. “I think it’s simplistic to say that the development team can ignore those issues on a forward-going basis. It’s not really a question of whether the domain expert will ever be able to automatically just push a button and have a beautiful, efficient embedded system,” he explains. “It’s more of a question of, Can a design team quickly look at the various ways of implementing a system that addresses the domain issues but also addresses the implementation issues, and get it done in as streamlined a fashion as possible?”

## FLOATING VERSUS FIXED

When Sanarus’ Stevens got started on the Visica 2 project, he knew that his hardware target was CompactRIO. Often, however, system architects and domain experts don’t know what target embedded software will ultimately run on. Tung explains that a system engineer working in Matlab might describe an algorithm in floating-point terms, but that algorithm might ultimately run on a fixed-point processor. “You are not going to just put the algorithm on the fixed-point processor and assume it’s



Figure 3 One of two BA609 tilt-rotor aircraft undergoes flight testing, flying in airplane mode over the Alps in northern Italy (courtesy Bell/Agusta Aerospace Co).

going to work,” he says. “Over the last five years or more, [we’ve] enabled the persons working in Matlab or Simulink to essentially say within that environment that they are going to work on a 16-bit processor” so that they evaluate the behavior of their algorithm as it will work on that fixed-point processor. “A person working 10 years ago in an ideal algorithm would find all kinds of late surprises,” Tung adds. “By being able to evaluate the fixed-point performance in the Matlab or Simulink environment, the domain expert is able to look at the implementation effects and determine the impact.”

### DOMAINS VERSUS SYSTEMS

Like NI’s Tanzillo, Tung distinguishes between domain experts and system engineers or architects. Although NI promotes LabView for both, Tung says that domain experts working at the algorithmic level more often use Matlab, whereas an increasing number of Simulink users are “multidomain in their perspective.” He cites a mechatronics example, in which a system includes

mechanical, electronic, and embedded-software components. “The system-level view needs to capture each of those domains to let the system-level engineer say, ‘Is that design going to meet my requirements?’ The engineer would then ask, ‘If I needed to tune something in a domain to achieve the system-level performance I need, should I tune it in the embedded software, by changing the mechanical design, or by changing electronic design?’ And the nice thing about the Simulink environment,” Tung says, “is that the persons working with the Simulink models are able to look at the system design from any or all of those domain perspectives, and they are able to make changes in one of the domains and perhaps loosen up the requirements in another domain. The Simulink environment traverses the gap ... between the person who is thinking about things from an algorithmic standpoint and the person who thinks about it from an implementation standpoint.”

Tung also notes that, apart from performing system design, users of high-level tools can leverage them to perform

verification. “Let’s take a case in automotive-suspension systems,” he says. “You’ll have the model of the suspension systems and the algorithms that will be describing the damping and other behaviors, but you will also have the portion of the models that will be describing road surfaces, vehicle dynamics, driver behavior, and all the other things important to understanding if the suspension will do what you want. One portion of the models that describes the suspension system can automatically generate the code that goes into the microcontroller, and it ships as part of the car. The other portion of the models ... becomes the basis for the test bed; you essentially generate code for that other portion of the model comprising the road surface, the vehicle dynamics, and the driver, and you run that [code] in a real-time HIL [hardware-in-the-loop] system that essentially is the test bed.”

### FLY BY WIRE AT BELL/AGUSTA

David King, principal engineer at Bell/Agusta Aerospace Co, also has opinions on trends with domain experts. “Look-

ing back 15 years when we did some of our legacy fly-by-wire development programs, we had a much larger proportion of specialists for low-level programming languages on our team,” he says. “Now, if we look at the proportion of the team, the majority of them are not well-versed in the low-level languages, but they are more or less system-level engineers using the model-based design process.” The shift seems to be gradual, he adds. “I don’t see that we are dropping our number of specialists in the programming languages, but we are growing on the systems side, so the proportion is changing,” bringing good news for embedded-system designers.

King is currently working on the Bell/Agusta BA609 aircraft project, the first commercial, nine-passenger, tilt-rotor vehicle (Figure 3). The 609 program provides some perspective. “We started with model-based design in 1998,” he says. “That was even before the term ‘model-based design’ was coined. We put together a process using Matlab and Simulink to try to take out some of the manual steps in the process, such as hand-coding from design data. And the one thing that’s interesting is it’s not just hand-coding for the embedded source code, but it was the hand-coding for all the analytical tools that analyze the various aspects of the aircraft and system performance,” including simulation, development of the test cases, and verification.

Bell still employs hand-coded embedded software in the flight-control computers on the aircraft but uses the auto-coded models for all the analytical tools to perform simulation, structural analysis, stability analysis, and test-case generation, all of which, King says, represent a big chunk of the workload. “I am really seeing the benefit of model-based design in that we can have one model of the aircraft and the flight-control system that is in a very usable format, and that model is used by various disciplines to do analysis.” He adds, “For example, we use it for simulation to analyze dynamic loads and to analyze handling qualities. We use it for real-time simulation with the pilot in the loop, and we also use it for non-real-time evaluation when we are checking stability and control characteristics.”



Figure 4 Domain experts—in this case, creative and usability specialists—can now contribute directly to the development of embedded products’ user interfaces by employing the drag-and-drop Nucleus Graphics Designer tool (courtesy Mentor Graphics).

In addition, the use of standard models in Matlab and Simulink makes it easier to work with partners, including the company that builds the flight computers, and eases certification. Now, one aircraft in Texas and one in Italy are undergoing flight testing, which is about 75% complete, in anticipation of the beginning of the certification process with the FAA (Federal Aviation

Administration) and the European Aviation Safety Agency. The Simulink pages become part of the software-design data for the FAA-mandated DO-178B design-assurance process for the software development. King expects the trend to continue toward an increasing emphasis on high-level tools. “If you look at the tool set and the maturity of the tools that The MathWorks produces and that the industry is using now, they’ve progressed quite a bit since we started [the 609] program 11 years ago,” he says. “For our next programs, we are looking at additional automated steps, including [generating the] embedded software.”

## PRODUCT DIFFERENTIATION

Mentor Graphics addresses the embedded-system market with several product lines, including the Nucleus RTOS, which, according to Geoff Kendall, product-marketing manager for the embedded-systems division at Mentor, is the most widely deployed commercial RTOS because of its penetration in the mobile-phone market. When it comes to domain expertise, however, the relevant tool is the Nucleus Graphics embedded UI (user-interface) engine and designer tool, and the relevant domain experts are graphics artists (figures 4 and 5). Kendall calls the Nucleus Graphics Designer tool a “framework that allows you to completely separate the de-



Figure 5 Artists have created two user interfaces with no coding or scripting using the drag-and-drop Nucleus Graphics Designer tool (courtesy Mentor Graphics).

sign and presentation of the user interface from the functions underneath it.” He describes the traditional design flow: “For example, you’ve got graphic designers who mock up concepts in Photoshop to show what the screen should look like [with respect to menus and so forth]. The way it has traditionally worked, a couple of guys sit there coming up with those concepts; then they hand them over to the engineers, and the engineers roll their eyes and say, ‘We’ll do our best, but we’ve got a bit of a crunch on.’ And you end up with a product that may be off to market on time and maybe slightly resembles what the graphic designers wanted, but it is probably not ideal for everyone.

“The domain experts here,” Kendall continues, “are the usability experts—in this case, the guys who know how to make a product work well and differentiate it from the other products on the shelf. But they are not the guys who really understand how to push pixels around the screen with an embedded processor.” One approach is to run a program that the graphics designers understand, such as Adobe Flash, on the target device. “Anyone who has tried to do that [task] will tell you Flash was never designed for the performance constraints of an embedded platform,” he says. “The exact attributes of Flash that made it so powerful and so dominant in the world of PCs and Web browsers are actual liabilities in the embedded space. For example, the reliance of Flash on vector graphics allows it to display on any screen size very well on a desktop PC, but that [ability] actually requires a lot more processing power to be able to render graphics in real time. So when you get down to an embedded device with a constrained screen size, you still have the computational hit of rendering those vector graphics. You actually start to see a significant performance falloff to the point that, if you use a native-UI technology like ours, you could be getting 15 or 20 frames of animation a second even on an ARM7 device. You would barely even get Flash running on that device at one or two frames per second.”

The Mentor Nucleus Graphics Designer tool, Kendall says, “allows peo-



**Figure 6** In conjunction with the Altium Designer unified-electronics-design tool, this daughterboard for its desktop NanoBoard reconfigurable-hardware-development platform targets Altera’s Cyclone III EP3C40 FPGA in a 780-bump BGA package (courtesy Altium).

ple with no programming or even any scripting [experience] to be able to drag and drop their UI designs to completely change and define the look and feel of a device, and they can do that in parallel with the engineers working on the coding with no interdependencies between the two whatsoever.”

With respect to product differentiation, he adds, the tool is “particularly relevant for guys who are building embedded products that will have multiple end customers. Imagine, for example, an in-car-entertainment system that a company is building for use by multiple automobile manufacturers, such as Ford, Daimler, and Ferrari.” These customers want different user experiences, but the supplier ideally wants one code base that never changes. The designer tool, says Kendall, supports that requirement without any recompilation of code.

➤ [Go to www.edn.com/090709cs](http://www.edn.com/090709cs) and click on Feedback Loop to post a comment on this article.

➤ For more technical articles, go to [www.edn.com/features](http://www.edn.com/features).

The concept extends far beyond automotive applications. “A number of big vendors of white goods actually have multiple consumer-facing brands,” Kendall explains, “and they need to make products that will look different for each of those brands. If they are doing that [task] by having different-shaped plastic dials and buttons, then it gets pretty expensive to do, and, if they start adding more features, they’ve got more and more buttons to add. It gets to the point where it actually becomes more cost-effective to start adding a screen to these devices. And we are now starting to see the likes of Samsung and Sony and big consumer-electronics companies starting to add touchscreen UIs even to things like fridges.” Mentor is well-positioned, he says, to support the “demand for adding interactive-touchscreen capability to devices that, two or three years ago, you would never have considered would need it.”

Tools such as Nucleus Graphics Designer are becoming more important as silicon becomes more complex. A lot of today’s processors that are going into devices with screens offer features such as Open GL 3-D hardware acceleration.

“This is all great stuff,” says Kendall. “But unless you’ve got software on top of it that lets you unlock the power, it’s just wasting space.” The designer tool doesn’t require hardware features, such as acceleration, but takes advantage of them if they are there, he explains. “It will allow the graphic designer to be able to do things like spin stuff onto the screen, fade it away—the kind of effects you’d normally expect to see in top TV production.” There has been a technical gap between what the graphic designers wanted to do and what the engineers had the time and experience to be able to build and deliver. The designer tool provides a layer of abstraction over the complex silicon to enable drag-and-drop graphical design to bridge that gap.

### CONTINUED EVOLUTION

Marty Hauff, master electronics-designer-success manager at Altium, expects to see continued evolution in the role of domain experts. His company supports the system-design process with its Altium Designer software for FPGA-populated PCBs; the company also offers related development boards (Figure 6). “As design abstraction increases, low-level hardware/software/microprocessor skills will become less important,” he says. “Innovation will move out of the hardware/software-design process and move into the overall user experience. A system-level designer could then be responsible for the specification and in-

novation of a system, and the low-level stuff would be farmed out as necessary.” Hauff likens what is happening in the embedded-system market to what has happened in the PC arena. “With tools such as VBA [Visual Basic for Applications] being embedded into Microsoft Office apps, nontraditional programmers can now create very advanced applications without needing to know too much about the underlying hardware,” he says. “And, with Web-based languages, such as Java and C#, software developers can now build applications with zero knowledge of the underlying hardware.”

Hauff doesn’t write off dedicated embedded-systems engineers, however: “I think it is probably a little too strong to say that domain experts will supplant engineers. I would argue that a design role is still required, but the skill necessary to fulfill that role will change. So it may well be that the nature of what engineers do changes. The need for low-level design engineers may be replaced by a greater need for system-level-integration engineers, and so on. This [sce-

nario] is consistent with what has happened to engineers in the past. As ICs have packed more functions, engineers can operate at a higher level of abstraction and create new products without needing to know as much low-level detail.”**EDN**

### REFERENCES

- 1 Nelson, Rick, “Engineers to be supplanted by domain experts?” *Test & Measurement World*, April 21, 2008, [www.tmworld.com/blog/640000064/post/280025228.html](http://www.tmworld.com/blog/640000064/post/280025228.html).
- 2 Totty, Michael, “The 2008 Technology Innovation Awards,” *The Wall Street Journal*, Sept 29, 2008, <http://online.wsj.com/article/SB122227003788371453.html>.



### FOR MORE INFORMATION

- |  |  |
|--|--|
| <b>Adobe</b><br><a href="http://www.adobe.com">www.adobe.com</a>   | <b>The MathWorks</b><br><a href="http://www.mathworks.com">www.mathworks.com</a>                               |
| <b>Alliance<br/>Spacesystems</b><br><a href="http://www.alliancespace&lt;br/&gt;systems.com">www.alliancespace<br/>systems.com</a> | <b>Mentor Graphics</b><br><a href="http://www.mentor.com">www.mentor.com</a>                                   |
| <b>Altera</b><br><a href="http://www.altera.com">www.altera.com</a>  | <b>Microsoft</b><br><a href="http://www.microsoft.com">www.microsoft.com</a>                                   |
| <b>Altium</b><br><a href="http://www.altium.com">www.altium.com</a>  | <b>National Aeronautics<br/>and Space<br/>Administration</b><br><a href="http://www.nasa.gov">www.nasa.gov</a> |
| <b>ARM</b><br><a href="http://www.arm.com">www.arm.com</a>   | <b>National Instruments</b><br><a href="http://www.ni.com">www.ni.com</a>                                      |
| <b>Bell/Agusta<br/>Aerospace Co</b><br><a href="http://www.bellagusta.com">www.bellagusta.com</a>                                  | <b>OpenGL</b><br><a href="http://www.opengl.org">www.opengl.org</a>  |
| <b>Cal-Bay Systems</b><br><a href="http://www.calbay.com">www.calbay.com</a>   | <b>Samsung</b><br><a href="http://www.samsung.com">www.samsung.com</a>   |
| <b>Embedded Systems<br/>Conference</b><br><a href="http://www.embedded.com">www.embedded.com</a>                                   | <b>Sanarus</b><br><a href="http://www.sanarus.com">www.sanarus.com</a>   |
|  | <b>Sony</b><br><a href="http://www.sony.com">www.sony.com</a>  |