EMBEDDED SYSTEMS ENGINEERING powered by EECatalog

February 16, 2016

Chip Design

Guide to EDA tools, semiconductor technologies and services

Successful Trajectory to Deliverable System: Q&A with Pentek Vice President and Co-Founder Rodger Hosking

By Anne Fisher, Managing Editor

Increasing complexity and new trends—FPGA devices with embedded processors; data converters that use JESD 204B gigabit serial interfaces, to name just two—continue to make life interesting for systems integrators.

EECatalog spoke recently with Rodger Hosking, vice president and co-founder, Pentek. Hosking holds more than 30 years' electronics industry experience and has authored hundreds of articles about software radio and digital signal processing. During our Q&A, he addressed, among other topics: the benefits of optimized FPGA configuration code; the reasons "everything" isn't optical; and how systems integrators can steer clear of a finger-pointing fracas.

EECatalog: Let's start with a quick overview of Pentek.

Rodger Hosking, Pentek

Rodger Hosking, Pentek: What Pentek has been doing throughout our 30 years in business is very tightly connecting A/D and D/A converters to digital signal processing elements for systems that need to acquire and process signals for radio, radar and communications. This tight coupling allows those wideband signals to be treated for the application in real time, whether it's a communications transceiver, airborne radar countermeasures system, signals intelligence monitor, or ground station for satellite telemetry—whatever processing needs to be done, you typically need to have an extremely fast, direct, dedicated connection between the data converters and the processing element.

EECatalog: Complexity in this industry is increasing. How has that changed how Pentek works with customers, if it has?

Hosking, Pentek: The overall concept of what we are doing is essentially the same, but you can be sure that every few months there's new technology around to allow you to do those things with greater speeds and greater processing power as well as smaller package sizes and higher densities.

Our customers can take advantage of the newest technology by buying the board-level products we provide because they are all built against open industry standards. This means system integrators can put these standard board-level products from several vendors into open architecture systems and develop their own custom software to meet the end user system-level requirements.

Because a systems integrator typically offers systems to his customers with a hard delivery date and a fixed price, we work with him to develop a successful trajectory for getting him from where he is to where he has a deliverable system up and running, on time and on budget. This can prevent a flaw in the system proposal discovered only during integration that might require twice as many boards as proposed, often forcing the system integrator to eat the cost of those extra boards.

So, it's critical for our customers to have the project fully vetted in the beginning to fully understand how the system will work and why it is likely to be successful. Because each system is unique, things are always going to be a little more complicated than you think, so you have to add some margin for little more complexity, a few more resources, and some extra speed in the interconnect paths between elements.

EECatalog: Experience can help with the fine-tuning.

Hosking, Pentek: Exactly. For example, an FPGA has a certain number of configurable logic resources. If you know that you are going to have to

use 96 percent of the resources in a proposed FPGA, you are probably going to be in trouble! You're much better off with a larger FPGA where you are using only 60 or 70 percent of its resources, because design surprises always pop up or a customer suddenly wants a new feature!

Same thing with transfer speed—if you are trying to move data right at the limit of the interface, you might want to consider using a faster interface that has 30 percent margin over what you really need just to add a comfortable cushion. Data traffic always tends to grow a bit when you get down into the real world.

EECatalog: Let's talk bit about Pentek's new FlexorSets. The release announcing the product named a number of features—which ones can most clearly trace their ancestry to conversations with embedded developers in such sectors as radar, communications and general data acquisition about what they needed to be more productive?

Hosking, Pentek: FlexorSet evolved for a very simple reason. The open architecture standard called FMC defines two different kinds of products. It defines a mezzanine card, which is the front-end module, usually containing data converters or digital interfaces. And it defines an FMC carrier, typically containing an FPGA or processor that hosts the FMC mezzanine module.

Because it's an open standard, you have vendors that make FMC modules and vendors that make the carriers. And each of the modules has a unique set of components for specific analog and digital interface functions. Each of the carriers has a unique complement of processors or different types of FPGAs, various form factors like VPX, PCIe, or CompactPCI, and different interfaces to the rest of the system,

Following this open standard to satisfy the needs of a specific application, a particular FMC module is installed on a particular carrier. However, the FPGA on the carrier must contain custom configuration code to match the characteristics of the FMC module. Each of these modules has a completely different personality, so each of them requires a different configuration of the FPGA in order to support its unique interface, and to make it work efficiently for optimum performance.

When you are getting a carrier from one vendor and a module from another vendor—who is going to create the custom FPGA code into the carrier to make the combination work? That task has often fallen into the lap of the system integrator. If there is a question or if he's struggling with trying to get them to work together, he often is faced with a vendor saying, "Well our module works fine, the problem is with the carrier vendor." The carrier vendor then says, "Our carrier works fine, it must be a problem with

the module." The system integrator is often caught in the middle of a finger-pointing contest. So, even though FMC is an open standard, developing configuration code for the FPGA is a significant consideration.

We offer both FMC carriers and FMC modules, and when we sell them together as FlexorSet, we install optimized FPGA configuration code so the customer doesn't have to develop it himself. He gets a module and a carrier as a completely integrated subsystem that will work right out of the box, fully supported with software drivers, high-level C function libraries, and example routines, all ready to run with no finger pointing! That's the advantage of FlexorSet.

EECatalog: Pentek has released a board compatible with the VITA 66.4—what will optical interfaces allow that copper does not?

Hosking, Pentek: The optical interface lets you operate with much higher data rates than you can over copper. If you have a critical high-speed link that needs to connect two boards within a card cage, say a 3U VPX chassis, or needs to go from one chassis to another chassis, optical links can be a very good way to handle it. Secondly, optical cables are extremely lightweight and thin, ideal for weight sensitive systems. Also, they are completely immune to the effects of electromagnetic radiation, interference and noise.

Optical also has the ability to go very long distances—hundreds of meters to kilometers—from an antenna on the mast of a Navy ship down to the engine room, for example. Signals being transmitted through that optical cable are not only immune to electrical noise from a huge collection of onboard equipment, but data is moving at a very high rate.

Lightweight, smaller optical cables are a major benefit for aircraft and UAVs, where weight is extremely critical. And, when there's an awful lot of electronics jammed into a very small space, eliminating susceptibility to magnetic radiation in cables is a big advantage.

So you ask, "Why isn't everything optical?" Well, it does still cost more money at this point in time to use an optical link. Since the rest of the system uses electrical signals, you have to convert between electrical and optical signals, and that takes a transceiver device with optical emitter and detectors, which requires not only board space, but also some extra power. So, you're paying a bit more in cost, space and power to gain some significant advantages of optical.

However, optical links may be the only practical option for some applications. They are well suited for systems where you need a very high-speed data interconnect, a very long transmission distance, very

small lightweight cable, or completely immunity from electromagnetic susceptibility and radiation. If one or more of those factors is critical, then the optical interface turns out to be a really good solution.

We have been shipping these optically connected systems now for about a year and a half, and we're getting a lot of interest. It's new, and yet, if you have the application drivers I've listed that make optical attractive, adopting these new open standards is a great way to get started. With multiple industry vendors offering a range of products, including connector and cable companies, board level vendors, optical transceiver vendors, and backplane and chassis suppliers, the industry infrastructure is already in place and growing. It has turned into an exciting and interesting embedded applications space.

EECatalog: Are there implications for security when we're discussing optical rather than copper?

Hosking, Pentek: Yes, in two ways. Let's say you have a very sensitive signal to send and you need to maintain complete security on its channel. You would probably start off by doing some sort of encryption or encoding of the signal. But, if you're using a copper cable, someone could place a very sensitive receiver next to that cable and eavesdrop on the electrical signal. If he's clever enough, he can decode or decrypt that signal and extract the information. With an optical cable, there is zero radiated energy, because nothing leaves this jacketed optical fiber cable. Tapping an optical cable is virtually impossible, because once you cut into an optical cable you compromise the signal so much that the breach is easily detected.

The second aspect of security is maintaining a high reliability link. Let's say you've got someone who wants to disturb or disrupt your communication. It would be possible for someone to use a high-powered noise generator near your copper cable to jam the signal or corrupt it. Of course, an optical cable is immune to that type of interference since its electromagnetic susceptibility is zero.

EECatalog: What's on your plate with regard to evolving FPGA technology?

Hosking, Pentek: Virtually all links in embedded systems are transitioning from parallel to serial, as evidenced in backplanes and in the data interfaces found in each new generation of CPUs, FPGAs and data converters. One of the things currently challenging us is interfacing FPGAs to the new class of data converters that use JESD 204B gigabit serial interfaces instead of the more traditional high-speed parallel LVDS lines.

We have to accommodate these new A/D and D/A devices with compatible interfaces in the FPGAs, but FPGAs are not yet fully up to speed in terms of making that connection straightforward. We have to work harder to develop new FPGA interface structures, so when our customers get a board from us, that thorny problem is solved and working. To make matters worse, each data converter presents its own special flavor of gigabit serial interfacing, so a unique solution is required for each. It is a big job and quite tricky, and it's just another example of the benefits of our FlexorSet offerings.

While FPGAs have had serial interfaces for some time, they have not had to deal with the specialized kind of serial interface that these data converters present, which are now becoming more mainstream. In future offerings, we expect to see additional support for these kinds of devices from FPGA vendors.

We are also seeing a lot of new FPGA devices with embedded processors, typically ARM Core processors. So, you now have a C language programmable processor and a high-performance FPGA together in the same chip, known as a System-on-Chip, or SoC. The concept has been around for a while, but we are seeing more customer interest and more offerings from FPGA vendors. With SoC, you really have two completely different programmable elements to deal with on a single piece of silicon. You have to have a C compiler and development tools for the ARM processor, as well as FPGA development tools for the FPGA logic. These tools are completely different in nature, and yet they have to work very closely together. This combination of two different animals in the same cage playing well together is a major challenge we are working on, and we will be announcing some new products very shortly.

These two trends we've seen just within the last year have made quite a significant difference in the way we are using and building products around FPGA technology.

EECatalog: Why are CPUs and FPGAs so different to program?

Hosking, Pentek: A lot of people think about FPGAs being programmed, and that is really not accurate. Computers and processors run programs. You write a program, typically in C language, and the processor executes those program instructions in sequential order exactly according to that script.

FPGAs are not processors, but rather configurable logic. Configuring FPGAs is really like building hardware, with you deciding how to connect the hardware resources together to perform a required task.

We find that our most successful FPGA designers often come from a hardware background, where they have designed circuit boards, or designed hardware systems, so they understand how hardware works and how it's put together. All you're doing in an FPGA is hooking up the bits of logic, registers, memory blocks, adders and multipliers that the FPGA contains, in the correct way to get the job done. Again, you're not really programming it, you're configuring it.

And the tools that you use to do programming in C and the tools that you use to configure an FPGA are quite different. The mindset of someone who is designing an FPGA has to be on hardware, and he has to be intimately aware of the hardware that he is building.

Software programmers correctly say, "I don't really care what the hardware is, I am writing a program that I can run on this computer or that computer." He has almost no visibility into the processor or the hardware that he is writing the software for. This makes it much easier to port code from one generation of processor to the next, a major benefit over FPGAs.

In embedded system design, you'll often encounter a certain critical function that you have to perform in your application, maybe a signal analysis or detection algorithm. You can achieve the function in either the CPU or the FPGA, now both on the same SoC device. Sometimes the choice is obvious and sometimes not. As a system designer you have to be competent enough to weigh the pros and cons of performance levels, design effort, schedule impact, maintainability and cost of resources. With SoCs, the added flexibility of having both CPU and FPGA resources available gives you more options, but also makes it more challenging to try to figure out how to solve each particular function.

http://eecatalog.com/chipdesign/2016/02/16/successful-trajectory-to-deliverable-system-qa-with-pentek-vice-president-and-co-founder-rodger-hosking/