

# Future-Proofing Military Applications Using FPGAs

Technology insertion using COTS FPGA-based products promises to deliver significant benefits but requires adequate planning.

*This article first appeared in Military Embedded Systems magazine, July 2007 (www.mil-embedded.com). Copyright OpenSystems Publishing. Used with permission.*

by Mark Littlefield  
Product Marketing Manager  
Curtiss-Wright Controls Embedded Computing  
mark.littlefield@curtiswright.com

Manuel Uhm  
Senior Marketing Manager, DSP  
Xilinx, Inc.  
manuel.uhm@xilinx.com

Over the past decade, the concept of “technology insertion” (the incremental upgrade of key technology components in a deployed or soon-to-be deployed system) has become a sort of mantra for the aerospace and defense (A&D) community, providing fielded systems with the latest, most advanced technology and the least delay. The rapid pace of technological innovation – and subsequent obsolescence – has made technology insertion critical for the success of multiyear A&D development, test, and deployment projects.

Unmanned aerial vehicles (UAVs), radar, and signals intelligence (SIGINT) are examples of sophisticated platforms that have benefited from technology insertion. To fully understand the chal-

lenges of technology insertion, including complexity and cost, it’s useful to consider what is involved in successfully using this approach for upgrading legacy systems. Using FPGAs in a reconfigurable computing application provides a good example of the benefits and challenges entailed in making technology insertion work. This example also enables us to examine some common issues associated with technology insertion, how COTS vendors can best help their customers address those problems, and how the overall technology insertion problem can be significantly eased by proper planning.

## Using FPGAs in Technology Insertion

The basic problem domains involved in technology insertion can be roughly categorized as:

- How the equipment physically connects (hardware)
- How the operating environment, system libraries and utilities, drivers, and middleware provide an infrastructure for applications (system software)

Using FPGA technology in an embedded multicomputing system provides an interesting case example because it straddles both the hardware and system software domains. FPGAs fall into the hardware domain because they are physically integrated to hardware elements; developers must work very closely with these elements when designing an application.

FPGAs present a software challenge as well, because they must be programmed and often incorporate vendor or third-party supplied blocks or IP. In addition, a general-purpose processor using system library calls typically configures the FPGAs and issues the commands they use to communicate with other processors and devices in the system. Such systems are often complex and heterogeneous, as shown in Figure 1.

For technology insertion, the ideal scenario is a plug replacement module that requires no hardware or software changes. Although this is uncommon for FPGA-based computing products, this scenario can also often be undesirable, as state-of-the-art computing platforms have new, more advanced features that system inte-

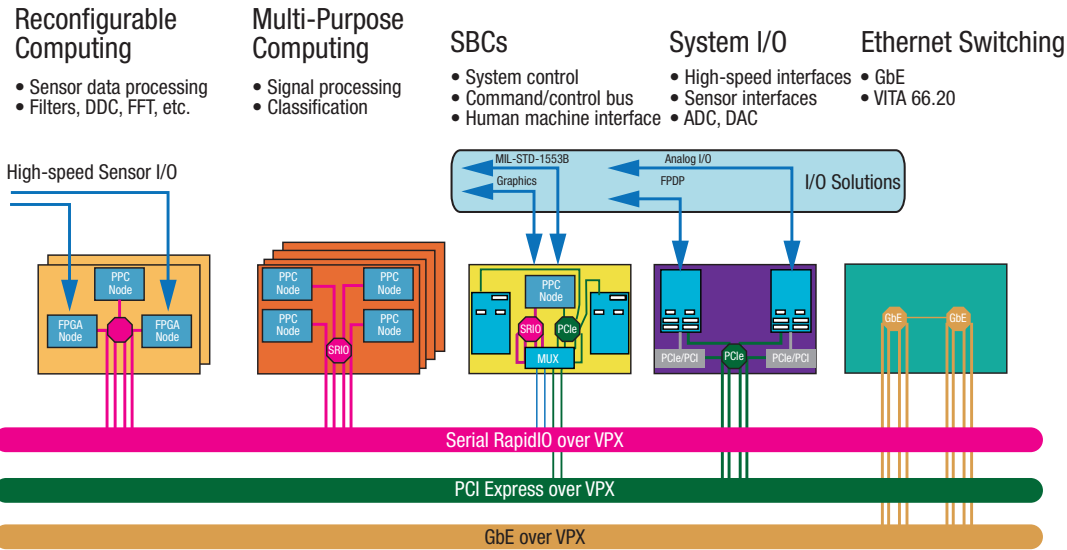


Figure 1 – High-performance embedded signal and image processing systems are often made up of numerous heterogeneous computer and I/O components connected by multiple communications fabrics.

grators can use to their advantage. Thus, the goal for system integrators must be to maximize the benefits of technology insertion while minimizing impact on the existing system. The degree of flexibility in the hardware and software domains can be directly affected by early design choices and products offered by COTS vendors.

The simplest approach for a COTS vendor to ease technology insertion hardware problems is to follow a common hardware model from platform generation to platform generation and to minimize connector pin changes. Industry board and module standards such as VME, VPX-REDI (VITA 46/48), PMC (VITA 32), and XMC (VITA 42) address a large part of this problem by specifying fixed form factors and pin definitions for board I/O such as buses or switched fabric interconnects. Vendors can extend this model by maintaining pin footprints across product generations for such common interfaces as serial ports and Ethernet. The same also holds true for FPGAs. A common, or at least similar, I/O footprint minimizes the need for radical redesigns during a technology insertion project.

### The Importance of Software

Although FPGA development is very tightly linked to the target component and platform hardware design, there is a lack of

standardized or industry-accepted tools and frameworks to abstract this linkage. The result is that software can typically have an even greater effect on a technology insertion program than hardware.

The software challenges can be somewhat mitigated on general-purpose processors by using off-the-shelf operating systems such as VxWorks or Linux, supported with full-featured BSPs, communications middleware, and application frameworks. For COTS vendors of FPGA products, the challenge is to provide development tools to integrators to ease technology insertion while maintaining the performance demanded by application developers.

A common approach is for COTS vendors to develop a standard “wrapper” or gasket that essentially represents the static infrastructure, such as interfaces to ADCs, memories, Ethernet, and so on. The blocks and infrastructure should be designed to support a set of commonly implemented use cases in the most efficient manner possible so as to minimize the size of the wrapper. Ideally, such an infrastructure would represent only 5 to 10 percent of the total die size of the FPGA.

### Software Support

No less important than the ease of integrating existing application code into a new FPGA platform for a technology insertion

project is the integration of the FPGA-based application into a larger multicomputer system. General-purpose computing elements are often closely tied to the system’s FPGAs by performing various command and control functions such as DMA engine control. Subsequently, the ease of technology insertion can be directly affected by how well a vendor can maintain APIs from one product generation to the next, which in turn is often linked to the level of abstraction in the API. The more abstraction, the less likely it is that the API will change between product generations. One of the key tasks of an external processor is the command and control of data movement, both within the FPGA and between the FPGA and other general-purpose processing nodes.

### Making Technology Insertion Real

Several COTS vendors offer common hardware strategies to aid technology insertion. One example is Curtiss-Wright’s latest generation of VPX-based board products. The three primary computing platforms in this product line – a dual 8641-based SBC, a quad 8641-based DSP engine, and the dual Xilinx® Virtex™-5 FPGA-based CHAMP-FX2 board – all share a set of common hardware design elements and were specifically designed with common pin footprints for common I/O elements. (See Figures 2 and 3.) For instance, the

FPGA board not only shares the PowerPC node design elements with the two other VPX boards, but closely resembles its preceding VME version in its I/O and memory configuration.



Figure 2 – High-performance FPGA families, such as the Xilinx Virtex-5 family, have common elements for logic, DSP, memory, and I/O, with a common package strategy enabling technology insertion at the component level.

The design of the VPX FPGA board and its VME predecessor took a balanced approach between the twin goals of maximum technology insertion value and minimum insertion effort. Curtiss-Wright's Continuum FXtools design kit provides a highly optimized set of IP blocks focused on peripheral I/O and memories, with only a lightweight, scalable switching block and a few additional utility blocks to ease application integration. By providing a minimal but highly optimized infrastructure, FXtools abstracts those common I/O and memory objects without sacrificing performance. Designing with these IP blocks and adopting basic use cases can help ease future technology insertion.



Figure 3 – Curtiss-Wright Controls CHAMP-FX2 Virtex-5-based VPX computing module

To ease both the integration and technology insertion of FPGA-based computing elements, the board's Continuum IPC communications middleware was extended to directly control DMA-based data-transfers involving the FPGA node. The IPC software enables application developers to create named buffer and data transfer objects that are globally visible to the system. Thus, any processor in the IPC system can create named buffers, attach to already created named buffers, and create data transfer objects between buffers without having to resort to code-manipulating, complex memory map translations or DMA command packet creation.

### Avoiding Technology Insertion Headaches

Although hardware and system software commonality across product generations is an important element in planning a successful technology insertion project, the application developer and system integrator can themselves play an important role in defining how difficult a future technology insertion project may be. For example, if the application developer bypasses vendor or OS APIs to directly manipulate hardware, the resulting code will be significantly less portable to future hardware platforms. However, application developers or system integrators can structure their application/system in a number of ways to minimize technology insertion headaches.

The first rule of designing for later technology insertion should be, "Don't fight the system software or OS." Most

software frameworks, such as a BSP, utility library, or communications middleware like Continuum IPC are designed with one or more basic use cases or design patterns in mind. One of the easiest ways for developers to introduce problems in a later technology insertion project is to bend the software framework to match their favorite design pattern. Doing so virtually ensures that later developers will have to bend things in a different way simply to make it work.

The same holds true for FPGA IP developers. Although implementation details and interfaces may shift from one product generation to the next, the basic design patterns usually do not. Using vendor-supplied IP blocks and data flows in the manner that the vendor originally intended helps minimize the amount of recoding needed for a technology insertion project.

Another beneficial technique that can be employed both for software and FPGA IP is to analyze the vendor-supplied APIs and block interfaces, identifying those interfaces that are likely to shift in future products and creating an abstraction layer or "shim" between the interfaces and the application code. A helpful clue is that if an API or block interface closely mirrors the underlying hardware, it is likely to shift between product generations.

Although such layers can add unnecessary code and performance delays to a system, the performance impact is often minimal, while the benefits for a later technology insertion project can be enormous. Changing a shim is much easier than changing multiple instances of an API call or IP block instantiation. It's important not to overuse this approach, however, as adding shims or abstraction layers complicates designs and can sometimes make debugging more difficult.

### Conclusion

Technology insertion can live up to its promise of future-proofing, but only if it is adequately planned for up front. Most COTS vendors have developed successive generations of products with this in mind. To capture this value, system integrators need to work closely with such vendors early in the design cycle. ●●●