Concurrent Engineering

A 5-Step Process for Accelerating Deployment of Embedded Subsystems

Prepared by Gregory Donahue
January 2013
The Concurrent Engineering Paradigm

There are three major challenges to overcome in the rapid development of embedded software:

- How can I develop software for hardware that won’t be available for several months?
- When it’s available, how do I get access to the limited supply?
- How do I reduce my time and cost to protect my software investment when processing technologies change?

Traditional design methodology starts with Systems Architects, progresses to Software Development and then to Hardware Implementation. The problem with this methodology is that it takes time — with complex defense systems, a lot of time.

With the growing use of Intel® processors in defense platforms, a much better methodology exits. Concurrent engineering enables both software and hardware development to happen in parallel, producing these benefits:

- Achieves faster time to deployment
- Provides ability to use the latest hardware technology without delaying software development
- Mitigates/manages risk at each stage of the design cycle
- Facilitates software development by multiple teams
- Protects investment in application software
  - Portability
  - Scalability
  - Decoupled from hardware implementation
  - Affordability

Customer Case Study

New Hardware Keeps Legacy Code Competitive

A major defense prime had a very successful system that was starting to fall behind the competition in terms of performance. This program had hundreds of thousands of lines of code. The prime wanted to port it over to new hardware so they could leverage the higher-performance serial RapidIO as their sensor connection fabric. The upgrade also required moving from PowerPC to Intel processors and to move the data onto PCIe™.

As partners, the customer and Mercury designed a new subsystem based on the rugged, high-performance OpenVPX standard and moved to an Intel Core i7 dual-core processor. While the new hardware subsystem was being built, the customer leveraged Virtual Multi-Computing (VMC) techniques to perform a methodical, multistage port of the legacy code. This enabled continual testing and verification of functionality and performance.

At completion, over 99 percent of the VMC code was ported and operational on the new open, Intel subsystem — a substantial preservation of investment in the customer’s application software.
The 5 Steps of Concurrent Engineering

**Step #1: Mathematical Modeling**
In concurrent engineering, a typical design cycle starts with mathematical modeling, often using tools such as MATLAB (see Fig. 1). Here, algorithm development and modeling of waveforms, application components and system data flow happens.

**Step #2: Virtual Multi-Computing Simulation**
The next step is to progress to Virtual Multi-Computing (VMC) simulation, where the application is run on generic hardware such as workstations and servers (see Fig. 2). The application software team can be developing new code, porting existing legacy code (by recompiling to Intel), or a mix of the two using processor resources such as processor type, clock frequency and GPUs that align well to target deployed hardware.

At this point, the final real-time, SWaP-optimized embedded hardware likely is in development and possibly not available yet. But software development can proceed using the target Intel processor while the deployable target hardware development occurs. Similar to any software development, you can use productivity tools to debug the application. Also, you can have multiple development workstations connected together to simulate a multicomputer.
Step #3: Multi-Computing Emulation
Next is multi-computing emulation. Here, the target silicon is used (Intel processors, FPGAs and GPUs) in a PC platform along with early integration with real fabric interconnects. Fabric support includes:

- Multiple 10, 20 and 40 Gbps Ethernet
- Serial RapidIO
- Infiniband
- Data Plane fabrics such as that available in OpenVPX™

You also have the ability to seamlessly migrate simulated code and legacy software. Your risk is again reduced and schedule shortened by running and debugging the application on target silicon and fabrics. You can accelerate technology readiness levels.

At this stage, risk is reduced by providing early insight into subsystem interactions that could be responsible for subsequent system errors.

Step #4: Lab Target
After emulation, the next step is to start integration on a lab system using ruggedized hardware for prototyping and demonstration. By this time, your software is roughly 90 percent functional.

Here, you can perform the final application debug along with performance tuning and regression testing. For this stage, Mercury offers a uniquely modular lab target hardware chassis. This flexible unit can be tailored to your particular needs. Options include:
• 3U and 6U
• Air cooled and conduction cooled (and hybrid)
• Backplane options (switched, meshed)
• Pre-configured and tested with Intel, PowerPC and GPGPU boards
• Pre-configured and tested with non-Mercury hardware, such as XMC interfaces or your own custom hardware
• Standard pluggable power supply options (VITA 62) or table-top power supply options

Step #5: Deployable Target
The last step is integrating into the final, deployable target. Here, the hardware is in the final package and form factor. Environmental testing such as DVT, EMI, thermal, shock and vibration can occur, leading to the final acceptance test.

Accelerated Deployment

Complex engineering development processes like those encountered in aerospace and defense applications result in long design cycles, often measured in years. However, because today’s government procurement is based on ever-changing warfighter needs, the rapid development and deployment of embedded systems is essential. The concurrent engineering methodology enables this acceleration while keeping investments at a minimum. In addition, you benefit from:

• Being able to implement the latest generation of hardware
• Preserving legacy software investments
• Enabling dispersed software teams to work together on a common system
• Identifying and addressing software issues early in the design cycle

For more information, visit mrcy.com/innovation.

Customer Case Study

Taking Quick Advantage of the Latest Hardware

Given their ability to deliver high performance per watt, GPGPUs are widely used in the gaming industry. Many aerospace engineers, however, do not have experience programming GPGPUs.

This was the case for a recent Mercury customer. While the lab system was being built, the developers used VMCs to port their original code over for use on the new GPGPUs, to get timing on their code, and to perform regression testing.

By leveraging VMCs, the developers significantly accelerated their learning curve and were able to port their original code quickly. In addition, since the lab hardware was being used in a remote lab, the developers didn’t have access to it. The VMCs enabled them to continue to add to their code base, thus minimizing the need for development time on the remote system.

To learn more, visit mrcy.com

Innovation That Matters, Echotek, EchoCore, Ensemble and MultiCore Plus are registered trademarks, and StreamDirect, Application Ready Subsystem and ARS are trademarks of Mercury Systems, Inc. Other products mentioned may be trademarks or registered trademarks of their respective holders. Mercury Systems, Inc. believes this information is accurate as of its publication date and is not responsible for any inadvertent errors. The information contained herein is subject to change without notice.

Copyright © 2013 Mercury Systems, Inc.